(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 October 2001 (25.10.2001)

PCT

(10) International Publication Number
WO 01/80176 A2

(51) International Patent Classification[7]: G06N 3/00

(21) International Application Number: PCT/BE01/00065

(22) International Filing Date: 13 April 2001 (13.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0008985.4 13 April 2000 (13.04.2000) GB
PCT/BE00/00099 30 August 2000 (30.08.2000) BE

(71) Applicant *(for all designated States except US)*: **SYNES NV** [BE/BE]; Technologielaan 11, B-3000 Leuven (BE).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **VAN HULLE, Marc** [BE/BE]; Hoefstadstraat 72, B-3600 Genk (BE).

(74) Agents: **BIRD, William, E.** et al.; Bird Goën & Co, Vilvoordsebaan 92, B-3020 Winksele (BE).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

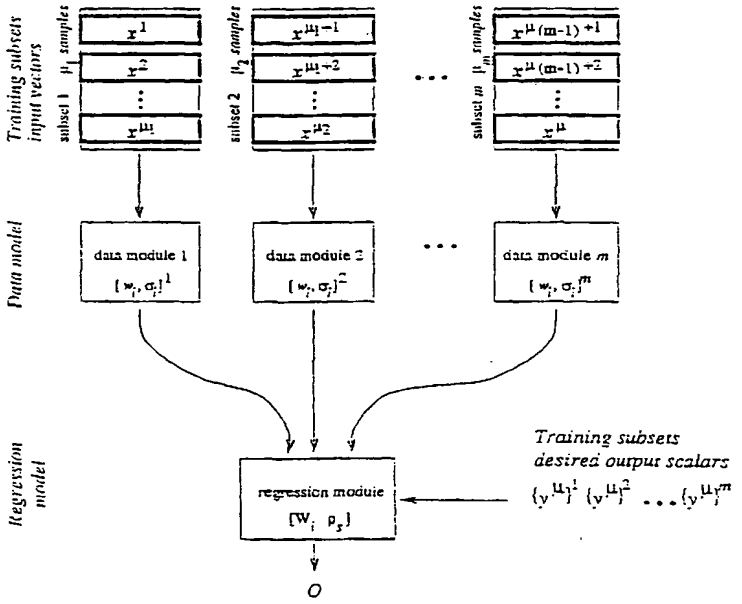
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:
— *of inventorship (Rule 4.17(iv)) for US only*

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHODS AND SYSTEMS FOR REGRESSION ANALYSIS OF MULTIDIMENSIONAL DATA SETS



(57) Abstract: The present invention may provide a method and system for distributed regression modeling. An initial or source data set is in a partitioned form, i.e. in one or more subsets, and data and regression modules are developed separately for these data subsets. The subsets can consist of either complete data vectors or incomplete data vectors, or of a mixture of the two. The data vectors may be incomplete, for example, since they contain missing vector components.

WO 01/80176 A2

METHODS AND SYSTEMS FOR REGRESSION ANALYSIS OF MULTIDIMENSIONAL DATA SETS

The present invention relates to systems and methods suitable for regression analysis of multi-dimensional data sets. The output of a regression model may be used for prediction, decision making, market analysis, fraud detection etc. The present invention relates to methods and systems, especially distributed processing systems as well as computer program products for regression analysis.

Technical Background

In non-parametric regression analysis, an unknown mathematical function f is modeled given a finite number of possibly inaccurate covariates or data points, without requiring any *a priori* assumptions other than perhaps a certain degree of smoothness. Hence, contrary to standard curve fitting procedures, no regression function parameters are optimized. The unknown function can be a scalar or a vector function. In the scalar (univariate) case, a function $y = f(\mathbf{x})$, with $\mathbf{x} \in V \subseteq \Re^d$, needs to be estimated from a given set of M possibly noisy input samples:

$$y^\mu = f(x^\mu) + \text{noise}, \quad \mu = 1,...,M, \tag{1}$$

where the noise contribution has zero mean and is independent from the input samples x^μ. In the vector (multivariate) case, the function $\mathbf{y} = f(\mathbf{x})$, with $\mathbf{y} \in \Re^{dy}$ is estimated as follows:

$$y^\mu = f(x^\mu) + \text{noise}, \quad \mu = 1,...,M. \tag{2}$$

The vector case is often treated as the extension of the scalar case by developing d_y scalar regression models independently, one for each vector component.

The regression performance improves if the number of "knots" in the model are not fixed but allowed to depend dynamically on the data points: indeed, "dynamic" knot allocation is known to yield a better regression performance than its static counterpart (Friedman and Silverman, 1989). Note that the "knots" are the points in V space that join piecewise smooth functions, such as splines, which act as interpolating

functions for generating values at intermediate positions. Alternatively, kernels are centered at these points (kernel-based regression).

Traditionally, kernel-based regression proceeds as follows. At each training sample $\mathbf{x}^\mu$ a kernel, such as a circular-symmetrical Gaussian, can be localized with a height equal to the corresponding desired output value y^μ; all Gaussians have the same radius (standard deviation) which, in fact, acts as a smoothing parameter. Formally, the output of the regression model, for a given input $\mathbf{x}$, can be written as:

$$O(\mathbf{x}) = \sum_{\mu=1}^{M} y^\mu \exp\left(\frac{-\|\mathbf{x}-\mathbf{x}^\mu\|^2}{2\sigma^2}\right) \qquad (3)$$

in the univariate (scalar) case, and

$$O_i(\mathbf{x}) = \sum_{\mu=1}^{M} y\_^\mu \exp\left(\frac{-\|\mathbf{x}-\mathbf{x}^\mu\|^2}{2\sigma^2}\right), \; i = 1, \ldots, d_y, \qquad (4)$$

in the multivariate (vectorial) case. In principle, σ is directly related to the noise distribution of the input signal however, such a measure is not always available. On the other hand, the radius can always be determined by cross-validation. The approach just described is adopted in the general regression neural network (GRNN) (Specht, 1991).

Alternatively, localisation can be restricted to only a limited number of kernels, say, N circular-symmetrical Gaussians. The kernels are normalized so that interpolation between the kernels centers, *e.g.*, using the normalized Gaussian can be carried out:

$$g_i(\mathbf{x}, \rho_x) = \frac{\exp\left(\frac{-\|\mathbf{x}-\mathbf{w}_i\|^2}{2\sigma^2}\right)}{\sum_{j=1}^{N} \exp\left(\frac{-\|\mathbf{x}-\mathbf{w}_j\|^2}{2\sigma^2}\right)} \qquad (5)$$

The output of the regression model for a given input $\mathbf{x}$ is then defined as:

$$O(\mathbf{x}) = \sum_{j=1}^{N} W_j g_j(\mathbf{x}, \rho_s) \tag{6}$$

in the univariate (scalar) case, and

$$O_i(\mathbf{x}) = \sum_{j=1}^{N} W_{ij} g_j(\mathbf{x}, \rho_s), \qquad i = 1, \ldots, d_y, \tag{7}$$

in the multivariate (vectorial) case. The W_j and W_{ij} terms are additional parameters of the respective regression models. The output of the univariate regression model is intended to approximate the desired input/output mapping:

$$O(\mathbf{x}^\mu) = \hat{y}^\mu \approx y^\mu = f(\mathbf{x}^\mu), \quad \forall \mu . \tag{8}$$

Similarly, the output vector of the multivariate regression model can be written as:

$$\mathbf{O}(\mathbf{x}^\mu) = \hat{\mathbf{y}}^\mu \approx \mathbf{y}^\mu = f(\mathbf{x}^\mu), \quad \forall \mu, \tag{9}$$

with $\mathbf{O} = [O_i]$. The positions (centers) $\mathbf{w}_i$ of the Gaussians are chosen in such a manner that the regression error, *i.e.*, the discrepancy between the output of the model and the desired output, for a given set of input/output pairs is minimal; the radii are chosen in an *ad hoc* manner. This is basically the Radial Basis Function (RBF) network approach introduced by Moody and Darken (1988). One could also extend this minimization of the regression error by including a common kernel radii σ, or even by including variable radii in this process (Poggio and Girosi, 1990).

In summary, the parameters of the kernel-based regression model, *i.e.*, the regression weights W_j and W_{ij}, kernel centers $\mathbf{w}_i$ and possibly also the common kernel radius σ or the variable radii σ_i, are optimized by minimizing the regression error. This is usually done by a learning algorithm which iteratively adjusts these parameters until the regression error becomes minimal (*e.g.*, for a separate test set of input/output pairs). This approach, however, provides optimization of the weights for a specific regression

application. For each new application the regression analysis has to be carried out on the source data.

It is an object of the present invention to overcome some of the problems of the known systems, in particular to provide a more flexible method and system of preparing data for regression analysis.

It is also an object of the present invention to reduce the amount of communication traffic when running regressions analysis on distributed data.

Summary of the present invention

The present invention may provide a method and system for distributed regression modeling. An initial or source data set is in a partitioned form, i.e. in one or more subsets, and data and regression modules are developed separately for these data subsets. The subsets can consist of either complete data vectors or incomplete data vectors, or of a mixture of the two. The data vectors may be incomplete, for example, since they contain missing vector components. This could be due to a partitioning into subspaces. The ensemble of the data modules form the data model and, similarly, the regression modules form the regression model. An advantage of the present invention is the minimization of the need for communication between the data and regression modules during their development as well as their subsequent use: in this way, delays due to communication are minimized and, since only model parameters need to be communicated, issues concerning data ownership and confidentiality are maximally respected.

The present invention may provide a computer implemented regression method for carrying out a regression application on a plurality of input data subsets of an input data signal, comprising the steps of:

developing for each of the input data subsets a data model independently of the regression application thus forming a set of data models;

generating a regression model for the input data signal using the set of data models. The developing step may include a map developing step in which an unsupervised, self-organizing, kernel-based, topographic map is developed from each input data subset by maximizing the entropy of the map's output, the topographic map being executed by a neural network; and during the development step receptive fields of the topographic map are truncated to receptive field regions, the receptive field

regions being independently scaleable in the space of the input data signal. The map may be equiprobabilistic. Preferably, at least some of the receptive field regions overlap. The receptive fields are preferably defined by the same kernel function. The receptive field regions are preferably hyper-spheroidal. Each data model may comprise data points and an interpolating function and the map developing step further comprises estimating a data density for each data point of a data model and the generating step includes the step of selecting a data point for the regression step based on the data density for that point. The developing step may comprise adapting the center location and the extent of each receptive field region to the input data density of the part of the input data signal data which activates the kernel thereof. The developing step may include use of any of the following methods: self-organizing maps (SOM), topographic maps, constrained topological mapping (CTM), transformation methods such as principal component analysis (PCA), independent component analysis (ICA), multidimensional scaling or similar methods, especially unsupervised methods. Each data record in the input data signal may be contained within one input data subset. However, each data record in the input data signal may be distributed over the input data subsets. Also some data records in the input data signal may be contained within one input data subset and some data records may be spread over some of the data subsets. The step of generating the regression model may comprise generating a regression sub-model for each of the data models, and combining the regression sub-models. The generating step may include use of any suitable regression analysis method such as multilayer perceptrons, (smoothing) splines, support vector machines (SVM), Radial Basis Function (RBF) networks, general regression neural network (GRNN), constrained topological mapping (CTM), Generative Topographic Map (GTM) and the mixture of experts network or similar.

The present invention may comprise a computer based system for executing a regression application on a plurality of input data subsets of an input data signal, comprising:

means for developing for each of the input data subsets a data model independently of the regression application to form a set of data models; and

means for generating a regression model for the input data signal using the set of data models. The means for developing may include a neural network in which an unsupervised, self-organizing, kernel-based, topographic map is developed from each

input data subset by maximizing the entropy of the map's output, and means for truncating receptive fields of the topographic map to receptive field regions, the receptive field regions being independently scaleable in the space of the input data signal. The means for generating the regression model may comprise means for generating a regression sub-model for each of the data models, and means for combining the regression sub-models. Means for generating a data density model from each data model may also be provided. Each data model may comprise data points and an interpolation function, and the system may further comprise decision means for selecting which data point of each data model is used for the generation of the regression model based on the output of the means for generating the data density model for that data point.

The present invention includes a computer program product for causing a processing engine to execute any of the methods according to the invention. The present invention also includes data carrier storing code segments of a computer program for executing any of the methods according to the present invention.

The present invention may make use of an unsupervised competitive learning rule for equiprobabilistic topographic map formation, called the kernel-based Maximum Entropy learning Rule (kMER) since it maximizes the information-theoretic entropy of the map's output as well as systems, especially distributed processing systems for carrying out the rule. Since kMER adapts not only the neuron weights but also the radii of the kernels centered at these weights, and since these radii are updated so that they model the local input density at convergence, these radii can be used directly, in variable kernel density estimation. The data density function at any neuron is assumed to be convex and a cluster of related data comprises one or more neurons. The data density function may have a single radius, e.g. a hypersphere.

The present invention may make use of a processing engine and a method for developing a kernel-based topographic map which is then used in data model-based applications. The receptive field of each kernel is disjunct from the others, i.e. overlapping. The engine may include a tool for self-organizing and unsupervised learning, a monitoring tool for maximising the degree of topology achieved, for example, by using the overlapping kernels, and a tool for automatically adjusting the kernel widths to achieve equiprobabilism. The receptive fields of the kernels may be convex, e.g. hyperspheroids or hyperspheres but the present invention is not limited

thereto.

The engines and methods described above may be local or distributed. The data model and the processing on the data model may be local or distributed.

By virtue of the topographic map learning rule, called the kernel-based Maximum Entropy learning Rule (kMER) all neurons of a neural network have an equal probability to be active (equiprobabilistic map) and, in addition, pilot density estimates are obtained By virtue of the learning rule, called the kernel-based Maximum Entropy learning Rule (kMER), pilot density estimates can be readily obtained since kMER is aimed at producing an equiprobabilistic topographic map.

The present invention will now be described with reference to the following drawings.

Brief Description of the Drawings

Figure 1 shows a distributed network with which the present invention can be used.

Figure 2 shows a further distributed network with which the present invention can be used.

Figure 3 shows embodiments of the present invention demonstrating Vertical modularity. (A) The data model and the regression model are developed separately and operate in sequence. (B) Several regression models can be grafted onto a common data model.

Figure 4 shows Kernel-based maximum entropy learning and the data model which can be used with embodiments of the present invention. (A) Definition of the type of formal neurons used. Shown are the activation regions S_i and S_j of neurons i and j (large circles) of a lattice A (not shown). The shaded area indicates the intersection between regions S_i and S_j (overlap). The receptive fields centers $\mathbf{w}$ and $\mathbf{w}_j$, are indicated with small open dots. (B) Neuron i has a localized receptive field kernel $K(\mathbf{x} - \mathbf{w}_i, \sigma_i)$, centered at $\mathbf{w}_i$ in input space $V \subseteq \Re^d$. The kernel radius corresponds with the radius of activation region S_i. The present input $\mathbf{x} \in V$ is indicated by the black dot and falls outside S_i.

Figure 5 shows an optimized kMER algorithm, batch version, adapted for the case where the values of some input vector components are missing (marked as "don't cares" or ×).

Figure 6 shows an algorithm for determining missing input vector components in accordance with an embodiment of the present invention.

Figure 7 shows a kernel-based regression of a function $y = f(\mathbf{x})$, $\mathbf{x} = [x_1, ..., x_d]$ using a two-step procedure: the data model and the regression model in accordance with an embodiment of the present invention. The data model comprises a topographic map of N neurons of which the activation regions are determined with kMER (location $\mathbf{w}_i$ and radii σ_i). The regression model consists of N Gaussian kernels, with the same location and (scaled) radii as the activation regions in the data model (*cf.* the icons next to g_1 and g_N). The outputs of these kernels, $g_1, ..., g_N$, are weighted by the parameters $W_1, ..., W_N$ so as to produce the regression model's output O. The $W_1, ..., W_N$ are trained with the delta rule in such a manner that the output O represents an estimate $\hat{y}$ of the function to be regressed, $y = f(\mathbf{x})$ (*cf.* the icon on the top).

Figure 8 shows embodiments of the present invention having horizontal modularity. Partitioning of the data set into subsets of the input space data set (A), or into subspace data sets (B), and a mixture of the two (C).

Figure 9 shows a further embodiment of the present invention having horizontal modularity. Case of several subsets and one regression module.

Figure 10 shows an embodiment of the present invention having multiple subsets and multiple regression modules. The latter are arranged at two levels.

Figure 11 shows an embodiment of the present invention having multiple subsets and regression modules and a single decision module.

Figure 12 shows an embodiment of the present invention having multiple subspaces and multiple regression modules at two levels.

Figure 13 shows an embodiment of the present invention involving a mixed case, heuristic strategy: multiple subsets of complete and incomplete input vectors. The presence of incomplete data vectors is ignored.

Figure 14 shows a further embodiment of the present invention involving a mixed case, correct strategy: the presence of complete data vectors is ignored.

6. Definitions

Data model: a lossy representation of main characteristics of a data set. The representations may be ones that reflect parameters of the density distribution, or

descriptors of its shape and extent, or the parameters of a projection of the data onto a subspace, or a transformation of the data such that a certain criterion is satisfied. The term "lossy" implies that the original data set is not contained within the data model and that the data model is a compacted representation in comparison with the source data set.

Equiprobabilistic: All units (neurons) in the lattice (neural network) have an equal probability to be active. All representational units should participate with the same probability in the representation. All kernels contribute equally in the density estimate provided by the topographic map (prior class probabilities are equal, if kernels would represent a different class).

Entropy: entropy of a variable is the average amount of information obtained by observing the values adopted by that variable. It may also be called the uncertainty of the variable.

Kernel-based: a particular type of receptive field – it has the shape of a local function, e.g. a function that adopts its maximal value at a certain point in the space in which it is developed, and gradually decreases with distance away from the maximum point.

Topographic map: a mapping between one space and another in which neighboring positions in the former space are mapped onto neighboring positions in the latter space. Also called topology-preserving or neighborhood-preserving mapping. If there is a mismatch in the dimensionality between the two spaces there is no exact definition of topology preservation and the definition is then restricted to the case where neighboring positions in the latter space code for neighboring positions in the former space (but not necessarily vice versa).

Receptive field: the area or region or domain in the input space within which a neuron (generally synonymous with synaptic element of a neural network) can be stimulated.

Self-organizing: refers to the genesis of globally ordered data structures out of local interactions.

Non-parametric density estimation: no prior knowledge is assumed about the nature or shape of the input data density. Non-parametric regression: no prior knowledge is assumed about the nature or shape of the function to be regressed.

Description of the illustrative embodiments

The present invention will be described with reference to certain embodiments and with reference to certain examples and to certain drawings but the present invention is not limited thereto but only by the claims. In particular the present invention will be described with reference to generating a data model using an equiprobabilistic, unsupervised topographic map but the present invention is not limited thereto. Other methods such as self-organizing maps (SOM), topographic maps, constrained topological mapping (CTM), transformation methods such as principal component analysis (PCA), independent component analysis (ICA), multidimensional scaling or similar methods, especially unsupervised methods. or similar may be used.

1. Distributed processing systems

Fig. 1 shows a basic network 10 parts or the whole of which may be used with embodiments of the present invention. It comprises two major subsystems which may be called data processing nodes 2 and data intelligence nodes 3, 4. A data processing node 2 comprises one or more microprocessors that have access to various amounts of data which may include very large amounts of data. The microprocessor(s) may be comprised in any suitable processing engine which has access to peripheral devices such as memory disks or tapes, printers, modems, visual display units or other processors. Such a processing engine may be a workstation, a personal computer, a main frame computer, for example or a program running on such devices e.g. a UNIX workstation or a personal computer with a Pentium III processor from Intel, USA and running Windows 98 operating system from Microsoft USA. The data may be available locally in a data store 1 or may be accessible over a network connection such as via the Internet, a company Intranet, a microwave link, a LAN or WAN, etc. The data may be structured such as is usually available in a data warehouse or may be "as is", that is unstructured provided it is stored in an electronically accessible form, e.g.

stored on hard discs in a digital or analogue format. The processing engine is provided with software programs for running an algorithm in accordance with the present invention to develop a data model which is independent of the regression application to be applied. The data model is preferably based on a topographic representation of the input data but the present invention is not limited thereto. Particularly preferred is a competitive learning algorithm capable of producing a topographic equiprobabalistic density representation (or map) of the input data having a linear mapping of real input data densities to the density represented by the topographic representation (the algorithm is described in more detail below). This topographic map may be described as a graph 7 of data models that represent the data processed. Each data model may be described as a compacted representation of the source data and comprises data points and a real function. The combination of the data points and the real function provide at least an approximate model of the source data. Each data model is stored on suitable storage medium and is a data structure in accordance with the present invention. The node 2 can use a persistent medium to store the graph of data models 7, or it can send it directly to other nodes in the network, e.g. nodes 3 and 4. Optionally, the data processing node 2 can be distributed over a network, for example a LAN or WAN. A data processing node 2 can run on most general purpose computer platforms, containing one or more processors, memory and (optional) physical storage. The supported computer platforms include (but are not limited to) PC's, UNIX servers and mainframes. Data processing nodes 2 can be interconnected with most existing and future communication links, including LAN and WAN systems. A data processing node 2 can also have a persistent storage system capable of storing all the information (data) that is been used to generate the graph 7 of data models that are stored or maintained by this node or a sub-graph of the graph 7. The graph 7 of data models can be saved regularly to the persistent medium for the analysis and monitoring of changes and the evolutions in the data (e.g. trend detection).

A data processing node 2 can also run other or additional algorithms that can be used to process data or pre-process or prepare data ready for analysis (e.g. to capture time dynamics in data sets). The data sets that are used can be both structured data (e.g. databases) or unstructured data (e.g. music samples, samples of pictures). The only limitation is that the data should be offered in a format that can be processed by the chosen computer platform as described above.

A data processing node 2 can provide a data intelligence node 3, 4 with an individual data model, a sub-graph or the complete graph 7 of data models. Normally, only the data models are returned not the data itself. However, it is (optionally) possible to return also the data that is used to generate the data models.

The graph of data models that is build up and maintained by a data processing node 2 contains:

1) A number of datanodes that contain the data models that describe at least a part of the data set.

2) A number of directed links between the datanodes.

Note: datanodes should not be confused with nodes of a distributed system such as 2, 3, 4 of Fig. 1. The word "Datanode" refers to a data model which models at least a portion of the input data to be processed. A datanode may be a software node. A datanode may be a neural network which models a part of a topographic equiprobabilistic density map, for example, a part generated by clustering after application of the novel competitive learning algorithm described above.

The datanodes and directed links are preferably organized in a hierarchical system, that is in a tree, in which topographic maps from two or more levels overlap.

There are a few special types of datanodes:

1) The top level datanode contains the data model from the complete data set and from this node, all the other datanodes in the tree can be reached from this top level via the directed links. It has only originating directed links - it has no terminating single directed link from another datanode.

2) A leaf datanode is a datanode that has no originating, single directed links to other datanodes. In a system that has gone through the complete initial training phase in accordance with the competitive learning algorithm in accordance with the present invention, this means that the data in this datanode has a substantially homogeneous distribution and that it is not relevant to split the data further. It can be said that a leaf node cannot be resolved into clusters which can be separated, that is it is "homogeneous", or any discernible clusters are of such minor data density difference that this difference lies below a threshold level.

All the other (intermediate) datanodes in the tree between the top datanode and a leaf datanode describe:

a) A subset of the complete dataset with common characteristics, but that can be refined further (i.e. without a uniform distribution).

b) The data model of the data described by this datanode.

During an initial training phase the following steps are taken which results in the formation of the datanode tree from top-down:

a) The top-level model is generated (i.e. generation of the top datanode).

b) This top-level model is divided in several parts in accordance with rules described in detail below. The division is not a simple tiling of the top level topographic map.

c) Each of these parts describe a subset of the complete dataset and a data model is generated for this subset, i.e. either intermediate or leaf datanodes or a mixture of the two is generated.

d) The data described by an intermediate datanode can be divided further into other intermediate datanodes and/or leaf datanodes. If it is not possible to divide an intermediate datanode further, this intermediate datanode is a leaf datanode.

After the initial run, the graph produced is a tree, from top-datanode to leaf datanode.

One of the additional advantages of the data processing nodes 2 in accordance with the present invention is the capability to distribute functions over the network 10. There are several advantages in distributing the data processing nodes across the network 10:

a) Higher performance. Although the algorithms in accordance with the present invention are capable of running on parallel computer systems as they are almost linearly scaleable, additional computing power may be advantageously provided by another computer.

b) Privacy: for example, the European privacy rules for personal information limit the transfer of personal information that is collected between companies. For data mining purposes, the characteristics of individuals are not usually important but the higher-level information or attributes of these individuals.

c) Integration or cost reasons. It is sometimes not feasible to install a complete data store, e.g. a data warehouse in one location.

The system in accordance with the present invention can fulfill all these requirements through a number of different distribution schemes. Several combinations of different distributed network schemes are possible of which two are described below with reference to Fig. 2 which shows a more complex network 20 than that of Fig. 1.

In a system 20 designed for master/slave processing of sub-graphs with common data source in accordance with an embodiment of the present invention, new data or data that is used to retrain the data models is provided to a master data processing node such as 15. Optionally, this node 15 retrains the main data model (top datanode in the graph of data models). In addition, it may determine to which intermediate datanode(s) this data point belongs and send it to the processor(s) responsible for this (these) intermediate datanode(s). Such an intermediate datanode can belong to another data processing node, somewhere else in the network 20, called a slave data processing node 12 - 14.

As the processing requirements increase, it is possible to add in additional slave data processing nodes such as 11 to the network 20. A slave processing node can also act as a master processing node for another slave processing node depending on the network configuration, e.g. in Fig. 2, the nodes 11, 12 are slaves of the master data processing node 13 on top of the data, but this node 13 is also in a slave relationship to the master node 15.

The data modeling methods of the present invention may be used with parallel processing. For example, a distributing master processing node 16 collects all the initial data and carries out top level clustering. It determines a plurality of independent clusters, e.g. two. It decides to process the first of the clusters and send the second cluster with its associated data to the processing node 15 for parallel processing of the data. Data updates will be received by the distributing master node 16 and the master node 16 determines if the data is to be processed by itself or by the alternative processing engine in node 15. To do this, the master node 16 keeps a mapping between the records and the relevant processing engine. For example, the parallel processing system may include a plurality of independent processing units each comprising a processor, local disk storage, and local memory. The independent processing units may be connected to a high performance switch which provides communications between the processing units. A controller work station is connected into the system for input of data and display of results. One such system is IBM Corporation's RISC System/6000

Scalable POWERparallel system.

In an alternative embodiment, the tree structure of the graph 7 of data models may be at least partly mapped to a hierarchical master/slave network 20 as described above. A specific embodiment of the use of a master/slave network will now be described. Initially, the complete graph 7 of data models is set up on every processing node. After this, the individual processing nodes assume their allotted task of processing their specific part of the tree. For this purpose they each receive the relevant data subset from the master processing node. From the other nodes in the system, each processing node receives the weighting factor updates for the neurons other than the ones the node processes. With the updated factors introduced into the graph 7, each node can process new data while accurately with the influence of other intermediate datanodes and leaf nodes being modeled correctly. As only weighting factors are shipped around the network and not data, the amount of signal traffic is much reduced. This aspect of the present invention is a direct consequence of the linear density mapping of the topographic map generated by the competitive learning algorithm in accordance with the present invention. Only if the data which is associated with a cluster or clusters can be separated cleanly and accurately from the total data is it possible to part process safely the graph 7 in a distributed way. If the data density estimation is only approximate, the labeling of data to be associated with a specific cluster/neurons is inaccurate. This means that a significant proportion of wrong data is shipped to each slave node (that is data which should not be contributing in the processing of the datanode processed on this processing node but should be contributing on another node). This falsifies the results at each node. As only the weighting factors are shipped around the system and these may be sensitive to the incorrect data, the whole graph 7 may become an incorrect decision making tool within a short period of time.

In accordance with a further embodiment of the present invention slave/slave processing of a complete graph without a common data source but with a centralization point (Master node) is provided. The total graph 7 of data models can be generated by several data processing nodes in a slave/slave configuration that have individual access to different (separated) data sets. In one embodiment the different slave data processing units process their own data and send the updates (revised neural weighting factors) regularly to a central data processing node, that unifies the different updates into a

single consistent data model. There are two types of distribution of data:

1) Horizontally distributed: the different datasets describe the same characteristics (i.e. columns in database table), but different subjects (i.e. different records in this table).

An example: the customer database of a multinational company will be distributed over several databases in the different countries, each database describing the same attributes but only of the local customers.

2) Vertically distributed: the different datasets describe different characteristics of the same (group of) subjects

An example: the salary database, the human resources database, the restaurant database all for the same employees of one company.

A combination of horizontal and vertical distributed data is also possible – a hybrid system in which a data record may be completely contained within one database or may be spread over some or all other databases. The way that a distributed network 20 as shown in Fig. 2 may be operated with vertically or horizontally distributed data form separate embodiments of the present invention. With horizontal distribution, each local processing node processes its own data. If it is necessary to query the whole data model, it is possible to devise queries which are sent around the network and collect the answers from each processing node and bring it to the questioning node, e.g. the master node. Schemes in accordance with the present invention which involve local processing of distributed data and local generation of the data model may be applied for cost or time-to-install reasons, i.e. to eliminate the need for collecting together a centralized data warehouse or for privacy reasons, if, for instance, the different data sets belong to different companies. This exemplifies an aspect of the present invention which is to leave data where it is and to only ship queries, answers or at most abstracted versions of the data (data models) around the network. For instance, an alternative embodiment involves generating a data model locally from the local data at one node and retrieving only the data models from other processing nodes. This would mean, in the example above that the data models from various countries would be brought together at one processing node. This would allow querying of all these models on a single computing platform. This may seem wasteful in data transfer in

comparison to only sending the query around the network and collecting the answers. However, there may be advantages in doing so - e.g. the answers and the queries may be kept more private or the query is of the type that all the data must be present at one location. Anyway, it is still less data transfer than collecting and updating a data warehouse.

It is also possible to separate some other data processing functions that are not really related to the generation and maintenance of the graph of data models. For example, a data processing node 16 can serve only as a device that collects the graph of data models from the other data processing nodes, and updates the data analysis nodes if required.

Data intelligence nodes provide the additional logic (called application components) needed to run the regression applications such as the following. A data intelligence node contains a processing engine and may be a workstation, a personal computer, a main frame computer, for example or a program running on such devices e.g. a UNIX workstation or a personal computer with a Pentium III processor from Intel, USA and running Windows 98 operating system from Microsoft USA. The data analysis system in accordance with embodiments of the present invention can be used in the following non-limiting list of regression applications: direct marketing, quality assurance, predictions, data analysis, fraud detection, optimizations, behavior analysis, decision support systems, trend detection and analysis, intelligent data filtering, intelligent splitting of data sets, data mining, outlier detection. Application components relate to programs run to analyze the graph 7 of data models or part of it to obtain a useful, concrete and tangible result. It is an aspect of the present invention that the data model generation is kept separate from the regression applications. When the regression application is partly mixed into the data model generation, the data model becomes restricted in its use. One aspect of the present invention is that the data model is as neutral as possible, i.e. it is not determined by the specifics of the data nor by the specifics of the regression application. This allows the data models in accordance with the present invention to have a longer useful life. New queries and new problems can be investigated on the data model. Due to the linear mapping of the real data densities into the topographic map, the data model is an accurate representation of the data despite being highly compressed in size.

There are basically two different data intelligence nodes (Fig. 2):

1) The Data intelligence node 17, 18 for machine-machine interaction is designed to offer data mining intelligence to other applications.

2) The Data intelligence node 4 for exploratory data mining applications, that allows a human analyst 5 to analyze data interactively.

A data Intelligence node 17, 18 for machine-machine interactions is a node containing one or more processors that has access to at least a sub-graph of the graph 7 of data models as generated by a data processing node. The node 17, 18 contains the predefined logic to execute a specific application as needed by another computer system. For this purpose, this node 17, 18 offers the possibility to answer queries relating to these specific applications through a number of standardized machine-to-machine interfaces, such as a database interface (ODBC/JDBC), middle-ware interfaces (CORBA, COM) and common exchange formats (text files, XML streams, HTML, SGML).

This node 17, 18 can be connected to a data processing server 16 that serves at least a sub-graph of the graph 7 of data models using any LAN or WAN connection. A constant connection is not required, a data intelligence node with a persistent storage system can store the (graph of) data models locally and an application component can be made available that can detect if the data model has to be resynchronized. A node 17, 18 can run on the same platform as a data processing node, it can run as a part of another application or it can run on small handheld devices (small computers, mobile phones). It is also possible to combine a data processing node and a data intelligence node on a single (physical) machine.

A data Intelligence node 4 for exploratory data mining (machine-man interface) is a node containing one or more processors that has access to a at least a sub-graph of the graph 7 of data models, analogue to a data intelligence node 17 18 for machine-machine interface, but this node 4 also requires a visualization device that allows the user to browse in the graph 7 of data models and the results provided by the data models. The user can analyze the data and run an application component for selected data (e.g. any of the application described above). Specific application components can help the analyst to detect specific behavior patterns.

It is possible to run most of the regression applications in the data intelligence

nodes in an interactive mode.

In accordance with an embodiment of the present invention a plurality of multidimensional source databases containing data to be processed are first prepared (if this necessary) and a data model is determined for each one, e.g. using a set of competitive unsupervised competitive learning rules (kMER). Optionally, a non-parametric density model is built up from the neuron weights and the radii obtained by kMER for each data model.

The present invention may be implemented on a computing system comprising computing devices e.g. a personal computers or work stations which have an input device for loading data. The computing devices are adapted to run software which carries out any of the methods in accordance with the present invention. The computer may be a server which is connected to a data communications transmission means such as the Internet, a Local Area Network or a Wide Area Network. A script file including the details of the data subsets of the data input signal may be sent from one or more near locations, e.g. terminals, to a remote, i.e. second location, at which the server resides. The server receives this data and outputs back along the communications line useful data to the near terminal, e.g. a result of the regression analysis.

Typically a computer has a video display terminal, a data input means such as a keyboard, and a graphic user interface indicating means or pointer such as a mouse. A computer 10 includes a Central Processing Unit ("CPU"), such as a conventional microprocessor of which a Pentium III processor supplied by Intel Corp. USA is only an example, and a number of other units interconnected via system bus. The computer includes at least one memory. Memory may include any of a variety of data storage devices known to the skilled person such as random-access memory ("RAM"), read-only memory ("ROM"), non-volatile read/write memory such as a hard disc as known to the skilled person. For example, the computer may further include random-access memory ("RAM"), read-only memory ("ROM"), as well as an optional display adapter for connecting system bus to an optional video display terminal, and an optional input/output (I/O) adapter for connecting peripheral devices (e.g., disk and tape drives) to system bus . The video display terminal can be the visual output of computer, which can be any suitable display device such as a CRT-based video display well-known in the art of computer hardware. However, with a portable or notebook-based computer, video display terminal can be replaced with a LCD-based or a gas plasma-based flat-

panel display. Computer further includes a user interface adapter for connecting a keyboard, mouse, optional speaker, as well as allowing optional physical value inputs from physical value capture devices of an external system which input the data signal. The devices may be any suitable equipment for capturing physical parameters.

A computer also includes a graphical user interface that resides within a machine-readable media to direct the operation of computer. Any suitable machine-readable media may retain the graphical user interface, such as a random access memory (RAM) , a read-only memory (ROM), a magnetic diskette, magnetic tape, or optical disk (the last three being located in disk and tape drives). Any suitable operating system and associated graphical user interface (e.g., Microsoft Windows) may direct the CPU. In addition, the computer includes a control program which resides within computer memory storage. The control program contains instructions that when executed on the CPU carry out the operations described with respect to the methods of the present invention. The instructions may be obtained by writing a computer program in a suitable language such as C or C++ for execution of any of the methods in accordance with the present invention and then compiling the program so that it executes on a computer.

Those skilled in the art will appreciate that the hardware may vary for specific applications. For example, other peripheral devices such as optical disk media, audio adapters, or chip programming devices, such as PAL or EPROM programming devices well-known in the art of computer hardware, and the like may be utilized in addition to or in place of the hardware already described.

The present invention also includes a computer program product (i.e. control program for executing methods in accordance with the present invention comprising instruction means in accordance with the present invention) can reside in computer storage. The instructions (e.g., computer readable code segments in storage) may be read from storage into the RAM. Execution of sequences of instructions contained in the RAM 24 causes the CPU to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software. Accordingly, the present invention may take the form of an

entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects.

Furthermore, the present invention may take the form of a data carrier medium (e.g. a computer program product on a computer-readable storage medium) carrying computer-readable program code segments embodied in the medium. The terms "carrier medium" and "computer-readable medium" as used herein refer to any medium that participates in providing instructions to a processor such as a CPU for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as a storage device which is part of mass storage. Volatile media includes dynamic memory such as RAM. Transmission media include coaxial cables, copper wire and fiber optics, including the wires that comprise a bus within a computer, such as bus. Transmission media can also take the form of acoustic or light waves, such as those generated during radio wave and infra-red data communications.

Common forms of computer-readable media include, for example a floppy disk, a flexible disk, a hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tapes, any other physical medium with patterns of holes, a RAM, a PROM, an EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereafter, or any other medium from which a computer can read.

These various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to a processor for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to the computer system can receive the data on the telephone line and use an infrared transmitter to convert the data to an infra-red signal. An infra-red detector coupled to a bus can receive the data carried in the infra-red signal and place the data on the bus. The bus carries data to main memory, from which a processor retrieves and executes the instructions. The instructions received by main memory may optionally be stored on a storage device either before or after execution by a processor. The instructions can also be transmitted via a carrier wave in a network, such as a LAN, a WAN, or the

Internet.

However, it is important that while the present invention has been, and will continue to be, that those skilled in the art will appreciate that the methods of the present invention are capable of being distributed as a program product in a variety of forms, and that the present invention applies equally regardless of the particular type of signal bearing media used to actually carry out the distribution. Examples of computer readable signal bearing media include: recordable type media such as floppy disks and CD ROMs and transmission type media such as digital and analogue communication links which may be used for downloading the computer program product.

With reference to Figs. 1 and 2 and the above description, the modular approach of the present invention will be described. This modular approach allows data models, data sub-models, regression models, regression sub-models and decision modules to be located on one computing device or distributed through a distributed network as described with reference to Figs. 1 and 2. Further, the various modules of the present invention may be implemented on a parallel processor architecture or on a multi-threaded single computer. With reference to the figures the plurality of independent data sets and data models may be located in a distributed fashion throughout a distributed network.

2. A modular approach

The present invention and its embodiment provides a modular approach to data modeling and regression. In essence, the data models are generated independently of the regression task. This is done with an optimization procedure that operates on samples $\mathbf{x}^{\mu}$ drawn from the input distribution (in V-space). This results in what is called a *data model*. Then, for a given regression task and, thus, for a given set of input/output pairs, only the regression weights W_j and W_{ij} are optimized so as to minimize the regression error; the data model parameters are kept constant. This second, *regression model* is specified by the regression application at hand. The data and regression models are, thus, developed separately, and they operate in sequence (vertically modularity, see Fig.3A).

It is particularly preferred if the data model is developed in accordance with a kernel-based topographic map formation procedure. As a result, the kernel centers $\mathbf{w}_i$ and kernel radii σ_i are obtained in such a manner that an equiprobabilistic map is

obtained: each kernel will be activated with equal probability and the map is a faithful representation of the input distribution. Other unsupervised data model generators may be used such as self-organizing maps (SOM), topographic maps, constrained topological mapping (CTM), transformation methods such as principal component analysis (PCA), independent component analysis (ICA), multidimensional scaling or similar methods, especially unsupervised methods.

The regression models may be, for example, multilayer perceptrons, (smoothing) splines, support vector machines (SVM), Radial Basis Function (RBF) networks, general regression neural network (GRNN), constrained topological mapping (CTM), Generative Topographic Map (GTM) and the mixture of experts network or similar

The present embodiment provides the following advantages:

- the regression surface will be locally more detailed when locally more samples are available and *vice versa*
- several regression applications can be grafted onto the same data model (Fig. 3 B)
- the set of input samples $\mathbf{x}^{\mu}$ can be split into subsets for which separate regression *modules* can be developed. The ensemble of these modules then forms the regression application (horizontal modularity, see Fig. 8A)
- the input space can be split into subspaces for which regression *modules* can be developed that together form the regression application (horizontal modularity, see Fig. 8B)
- the input samples $\mathbf{x}^{\mu}$ may be incomplete, *i.e.*, some of the vector components may be missing (incomplete data handling)

Hence the present embodiment is modular in two ways: vertically modular (data model and regression model) and horizontally modular (data modules and regression modules).

3.0 Vertical modularity embodiment

3.1.0 Data model

The data model consists of a lattice A, with a regular and fixed topology, of arbitrary dimensionality d_A, in d-dimensional input space $V \subseteq \Re^d$. To each of the N nodes of the lattice corresponds a formal neuron which possesses, in addition to the traditional weight vector $\mathbf{w}_i$, a hypervolume activation region which may be circular- or

hyperspherical, hyper-oval or similar in shape. Such an activation region S_i, has a radius σ_i, in V-space (Fig. 4A). The neural activation state is represented by the code membership function:

$$\mathbb{1}_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in S_i \\ 0 & \text{if } \mathbf{x} \notin S_i. \end{cases} \tag{10}$$

with $\mathbf{x} \in V$. Depending on $\mathbb{1}_i$, the weights $\mathbf{w}_i$ are adapted so as to produce a topology-preserving mapping: neighboring neurons in the lattice code for neighboring positions in V-space. The radii σ_i are adapted so as to produce a lattice of which the neurons have an equal probability to be active (equiprobabilistic map), *i.e.*,

$P(\mathbb{1}_i(\mathbf{x}) = 1) = \frac{\rho}{N}$, $\forall i$, with ρ a scale factor.

3.1.1. Training the data model

The data model is trained as follows. A training set $\mathrm{M} = \{\mathbf{x}^\mu\}$ of M input samples is considered. In batch mode, the kernel-based Maximum Entropy learning Rule (kMER) updates the neuron weights $\mathbf{w}_i$ and radii σ_i as follows (Van Hulle, 1998, 1999b):

$$\Delta \mathbf{w}_i = \eta \sum_{\mathbf{x}^\mu \in \mathcal{M}} \sum_{j \in A} \Lambda(i, j, \sigma_\Lambda)\, \Xi_j(x^\mu)\, Sgn(\mathbf{x}^\mu - \mathbf{w}_i), \quad \text{and,} \tag{11}$$

$$\Delta \sigma_i = \eta \sum_{\mathbf{x}^\mu \in \mathcal{M}} \left(\frac{\rho_r}{N} \left(1 - \mathbb{1}_i(\mathbf{x}^\mu)\right) - \mathbb{1}_i(\mathbf{x}^\mu) \right), \quad \forall i, \tag{12}$$

with $\rho_r \stackrel{\Delta}{=} \frac{\rho N}{N - \rho}$, and $\Lambda(.)$ the usual neighborhood function, *e.g.*, a Gaussian, of which the range σ_Λ is gradually decreased during learning ("cooling"), and Ξ_i the fuzzy code membership function of neuron i:

$$\Xi_i(\mathbf{x}) = \frac{\mathbb{1}_i(\mathbf{x})}{\sum_{k \in A} \mathbb{1}_k(\mathbf{x})}, \quad \forall i \in A, \tag{13}$$

so that $0 \le \Xi_i(\mathbf{x}) \le 1$ and $\sum_i \Xi_i(\mathbf{x}) = 1$. $\rho_r > 1$ is used so that the activation regions overlap. Finally, the previous batch learning scheme is also available for incremental learning. The skilled person in the art of neural network learning is aware of batch or incremental learning techniques.

The characteristics of the trained data model can be summarized as follows:

1. the lattice forms a discrete estimate to a possibly non-linear data manifold in the input space V
2. the lattice defines a topology-preserving mapping from input to lattice space: neighboring neurons in the lattice code for neighboring positions in V-space
3. more neurons will be allocated at high density regions in V-space ("faithful" map, Van Hulle, 1999b).

3.1.2. Incomplete data handling

The input data $\mathbf{x} = [x_i]$ used for training could be incomplete: for some vector components there is no value available. These vector components will be treated as "don't cares", $x_j = \times$. In particular, the active neurons are determined and the weights and radii are updated only by using the input vector components for which a value is available. For example, an adapted kMER algorithm is given in Fig. 5, in batch mode, but it can equally well be shown for incremental learning.

Filling-in missing entries

Once the data model is developed, the missing entries can be filled in by applying the incomplete input vector to the map, the neuron with the closest weight vector can be determined, by ignoring the missing vector components, and the latter can be replaced by those of the closest weight vector. The algorithm is shown in Fig. 6. Instead of using only the closest weight vector, all active neurons could be selected, and their weight vectors could be used for deriving estimates for the missing input vector components. One way to do this is by determining the average of the weight vectors, and then perform the substitution using this averaged vector. The neuron radii could even be invoked to determine the average. For example, if vector component v_j is missing, then the following can be done:

$$v_j^{\mu} \Leftarrow \frac{\sum_{i \in S_a} \frac{w_{ij}}{\sigma_i^d}}{\sum_{i \in S_a} \frac{1}{\sigma_i^d}}, \tag{14}$$

with S_a the set of the active neurons, and d the dimensionality of the input space. Still another way is to do the following:

$$v_j^{\mu} \Leftarrow \frac{\sum_{i \in S_a} w_{ij} p(\mathbf{w}_i)}{\sum_{i \in S_a} p(\mathbf{w}_i)}, \tag{15}$$

with $p(\mathbf{w}_i)$ the density (estimate) at position $\mathbf{w}_i$ in V-space (for the use of density estimates, see below).

3.2 Regression application

When the topographic map A is developed, thus, without using information about the desired outputs, the circular (spherical, hyperspherical) activation region S_i of each neuron can be supplemented with a kernel, *e.g.*, a *normalized* Gaussian:

$$g_i(\mathbf{x}, \rho_s) = \frac{\exp\left(\frac{-\|\mathbf{x}-\mathbf{w}_i\|^2}{2(\rho_s \cdot \sigma_i)^2}\right)}{\sum_{j=1}^{N} \exp\left(\frac{-\|\mathbf{x}-\mathbf{w}_j\|^2}{2(\rho_s \cdot \sigma_j)^2}\right)}, \quad \forall i \in A, \tag{16}$$

with ρ_s a smoothing factor. These kernels are then used as the neurons' output functions. The output of the regression model for a given input $\mathbf{x}$ can be written as follows:

$$O(\mathbf{x}) = \sum_{j=1}^{N} W_j \; g_j(\mathbf{x}, \rho_s) \tag{17}$$

in the univariate (scalar) case, and

$$O_i(\mathbf{x}) = \sum_{j=1}^{N} W_{ij} \; g_j(\mathbf{x}, \rho_s), \qquad i = 1, ..., d_y, \tag{18}$$

in the multivariate (vectorial) case.

Hence, there are two types of unknown variables in the regression model: the weights W_j, $\forall j$, or W_{ij}, $\forall i,j$, in the univariate or the multivariate case, respectively, and the smoothing factor ρ_s.

3.2.1. Training the regression application

Several types of algorithms are included within the scope of the present invention. In the following the simplest one is described, based on gradient descent, but the present invention is not limited thereto. A set of input/output pairs is considered, *i.e.*, the training set: $\mathcal{M} = \left\{(y^\mu, \mathbf{x}^\mu) \mid \mu = 1, ..., M\right\}$. Training the weights W_j is performed with the *delta rule*:

$$\Delta W_j = \eta\left(y^\mu - W_j g_j(\mathbf{x}^\mu, \rho_s)\right) g_j(\mathbf{x}^\mu, \rho_s), \tag{19}$$

in the case of univariate regression. The rule is readily extendible to the multivariate case by developing one series of weights for each output vector component. The following training set is considered: $\mathcal{M} = \{(y^\mu, \mathbf{x}^\mu) \mid \mu = 1, ..., M\}$, with $\mathbf{y}^\mu = [y_i^\mu]$. The delta rule now becomes:

$$\Delta W_{ij} = \eta\left(y_i^\mu - W_{ij} g_j(\mathbf{x}^\mu, \rho_s)\right) g_j(\mathbf{x}^\mu, \rho_s), \tag{20}$$

Projection pursuit regression

Another embodiment of the present invention applies projection pursuit regression (PPR) (Friedman, J.H., and Stuetzle, W. (1981) Projection pursuit regression. *J. Am. Statist. Assoc.* 76 (376), 817-823) to kernel-based regression with topographic maps. In PPR, the *d*-dimensional data points are interpreted through optimally-chosen lower dimensional projections; the "pursuit" part refers to an optimization with respect to these projection directions. For simplicity, the case will be considered where the function f is approximated by a sum of scalar regression models O_k:

$$O(\mathbf{x}^\mu) \doteq \hat{y}^\mu = \sum_{k=1}^{K} O_k(\mathbf{a}_k \mathbf{x}^{\mu T}), \tag{21}$$

with $\mathbf{a}_k$ the projection directions (unit vectors) and where T stands for transpose. Each output O_k refers a regression model of the type given in eq. (18). The parameters of O_k and projections $\mathbf{a}_k$ are estimated sequentially in order to minimize the mean squared error (MSE) of the residuals:

$$C(\mathbf{a}_k) = \frac{1}{M} \sum_{\mu=1}^{M} \left[O_k(\mathbf{a}_k \mathbf{x}^{mT}) - \left\{ y^\mu - \sum_{k'=1}^{k-1} O_{k'}(\mathbf{a}_{k'} \mathbf{x}^{mT}) \right\} \right]^2, \tag{22}$$

and where the term between the curly brackets denotes the kth residual of the μth data point, r_k^μ, with $r_1^\mu = y^\mu$. In order to further optimize the projection directions, *backfitting* can be applied: $C(\mathbf{a}_K)$ is cyclically minimized for the residuals of projection direction k until there is little or no change.

Determining the smoothing factor

Training for the smoothing factor ρ_s can also be done with the delta rule, however, better results (*i.e.*, more reliable than would be achieved by learning) can be obtained in the following manner. The following quantity is defined as the training error:

$$\mathrm{TE} = \frac{1}{M}\sum_{\mu=1}^{M}\left\|\mathbf{y}^{\mu} - \mathbf{O}(x^{\mu})\right\|, \tag{23}$$

with $\mathbf{O} = [O_i]$, with O_i as in eq. (18). In a modification of this method a test set rather than a training set is used, and the test error is determined rather than the training error. The decision depends on how many training samples are being used, and how many are available for testing.

TE is plotted as a function of ρ_s, and the ρ_s-value that minimizes TE is sought. A clear parabolic (TE, ρ_s) plot is expected. Only three points are then necessary to locate the minimum, theoretically, but the effect of numerical errors must be considered.

For example, the following strategy can be adopted for determining three TEs (assuming that $\rho_r > 1$ in kMER so that the activation regions overlap):

1. starting with $\rho_s = 1$, the regression model is trained and the first training error, TE^1, is determined
2. ρ_s is increased, *e.g.*, to 1.1, the regression model is trained again and the second training error, TE^2, is determined
3. ρ_s is decreased, *e.g.*, to 0.9, the regression model is trained again and the third training error, TE^3, is determined.

From the relative magnitudes of TE^1, TE^2, TE^3, the location can be estimated, or the direction where the minimum should be sought determined.

3.2.2. Summary

The characteristics of the present regression embodiment can be summarized as follows:

1. the data model can be developed separately from the regression model: it is simply an add-on of the data model;
2. training the regression model will be fast since only one stage ("layer") of connection weights need to be trained;
3. the regression procedure adapts itself to the data (self-organization): since with kMER the weight density will be proportional to the input density, the regression model will be optimal in the sense that it will be locally more detailed when there are locally more input data available (higher density in **x**-space), and *vice-versa*;

4. by virtue of the delta rule and the smoothness procedure, regression modeling is easily implementable.

4.0 Horizontal modularity embodiment

An important feature of the kernel-based approach is that the regression model does not extrapolate beyond the range of its outermost kernel (which can be easily detected): a zero output will be obtained. This enables to develop a horizontally-modular approach to regression and, thus, to exploit parallelism. Basically, there are two ways to look at this type of modularity: the data can be partitioned into subsets of the input space data set or into subspace data sets (Fig. 8A, B). Evidently, also mixtures of the two can be considered (Fig. 8C).

4.1. Subsets of data – single regression module

Assume m subsets of input/output pairs are available (see "Training subsets input vectors" and "Training subsets desired output scalars" in Fig. 9). Each subset of input vectors is used for developing a corresponding data model (data model 1, ..., m). The weights $\mathbf{w}_i$ and radii σ_i available in the m data models are then used for introducing kernels g_i in the regression module. The denominator in the normalized Gaussians definition refers to all kernels of all data models. Finally, the W_is and ρ_s are determined, across the m data models, according to the scalar regression task.

4.1.1. Vectorial regression task

The former can be extended to a vectorial regression task, and desired output vectors can be considered instead of scalars. d_y regression models are developed, one for each output vector component. This can be done in parallel.

4.2. Subsets of data – multiple regression modules

The previous regression modeling strategy can be extended by breaking up the single regression module into $m + 1$ regression modules at two levels (Fig. 10): at the first level, there are m regression modules, one for each subset of input/output pairs, and at the second level, there is one module that integrates the outputs of the first level modules in order to produce the final regression output O. The m level 1 regression modules are trained as before. For the level 2 module, the following weighted sum of

the outputs is taken as the level 1 regression modules:

$$O(\mathbf{x}) = \sum_{j=1}^{m} WO_j \, O_j(\mathbf{x}, \rho_s). \tag{24}$$

Again, the WO_j are parameters that can be trained in the same way as explained before, or in a similar way. If the subsets are perfectly disjunct, then one can simply take the sum of all O_j.

Bayesian approach

Another embodiment that does not require additional training will now be introduced: it will take advantage of the fact that the regression surface developed in each module will be more detailed when more samples are available. Hence, if there is disposed of a density estimate in each module, then there will be more certainty of the regression output produced by a given module if the corresponding input also corresponds to a high local density.

Rather than having a common second level regression module as in Fig. 10, which requires global learning, a common decision module is introduced, which does not require additional training (Fig. 11). It goes as follows. Let $p_j(\mathbf{x} \mid j)$ be the (conditional or component) density estimate of the jth module for input $\mathbf{x}$. The regression output produced by the module, i^*, for which the product of the conditional density and the prior probability is the largest is taken:

$$i^* = \arg\min_j \left(p_j(\mathbf{x} \mid j) P(j) \right), \tag{25}$$

with $P(j)$ the (prior) probability of the jth module. Unless disposing of prior knowledge, all $P(j)$s can be assumed to be equal to $1/m$, hence:

$$i^* = \arg\min_j p_j(\mathbf{x} \mid j). \tag{26}$$

For both cases, the regression output O becomes:

$$O(\mathbf{x}) = O_{i^*}(\mathbf{x}). \tag{27}$$

In other words, the regression output is taken for which the largest number of local input/output pairs were available for training. This selection criterion is reminiscent of a Bayesian pattern classification procedure (whence the procedure's name). The selection itself occurs in the decision module. An estimate of the probability density of each module is obtained from the data modules is explained in the articles Van Hulle, M.M. (1998), Kernel-based equiprobabilistic topographic map formation. *Neural*

Computat., **10**(7), 1847-1871, Van Hulle, M.M. (1999a), Hill-climbing, density-based clustering and equiprobabilistic topographic maps, *Internal Report Synes*, Van Hulle, M.M. (1999b), Faithful representations with topographic maps. *Neural Networks*, **12**, 803-823, in the book "Faithfull representations and topographic maps" Marc van Hulle, John Wiley & Sons, 2000 and co-pending International Patent Application PCT/BE00/00099 which is incorporated herein by reference.

An alternative is to interpolate between the regression outputs produced by all modules according to their density estimates. The regression application's output O can then be written as follows:

$$O(\mathbf{x}) = \frac{\sum_{j=1}^{m} O_j(\mathbf{x}, \rho_s) p_j(\mathbf{x}|j) P(j)}{\sum_{j=1}^{m} p_j(\mathbf{x}|j) P(j)} . \qquad (28)$$

Bayesian approach – Risk minimization

Maximizing the product of the conditional density and the prior probability might not be appropriate for some applications. For example, when the consequence of erroneously taking the output of module i is much higher than that of module j. In order to take such effects into account, one can define a loss matrix $L = [L_{ij}]$ of which each element L_{ij} is the loss incurred by erroneously taking module j when actually module i should have been taken. The risk is then minimized when:

$$\sum_{k=1}^{c} L_{ki^*} p(\mathbf{x}|i^*) P(i^*) < \sum_{k=1}^{c} L_{kj} p(\mathbf{x}|j) P(j), \qquad \forall j \neq i^*. \qquad (29)$$

The L_{ij}s can be based on experience or expressed in a more quantitative manner, for example, in monetary terms.

Advantages

The advantages of the present embodiment can be summarized as follows:

1. No additional training is required, a feature which is particularly important when dealing with regression modules that are developed on separate computers (data servers): in this way, no global communication between the data servers is needed in order to build a distributed regression application. In the more traditional procedure, a global communication of the training samples would be required.
2. A new set of data that one wishes to consider in the regression application can be included without retraining the whole application: the new data set can be regarded as a subset for which a separate data module and regression module are developed. This

can be done in parallel, thus, while the original regression application is in use.

3. Each regression module can be supplemented with a risk factor so that risk minimization can be performed when considering the outputs of the regression modules.

4.2.1. Vectorial regression task

The previous embodiment can be extended to a vectorial regression application, thus with desired d_y-dimensional output vectors, rather than scalars. d_y regression models are developed in parallel, one for each output vector component. All training processes can run in parallel.

4.3. Subspaces – multiple regression modules

4.3.1. Scalar regression task

The complementary problem is now considered: the data set is partitioned along m disjunct subspaces. There is first focused on the input vectors. The input space of the data set is partitioned along m subspaces of which the dimensionalities are d_i, $0 \leq d_i \leq d$, $\sum_1^m d_i = d$. For example, if input sample $\mathbf{x} = (x_1, ..., x_d)$ is considered, then data module 1 is developed using the subspace vector $(x_1, ..., x_{d_1})$, data module 2 using $(x_{d_1+1}, ..., x_{d_2})$, and finally, data module m using $(x_{d_{m-1}+1}, ..., x_{d_m})$.

In the event that the input data for these subspaces are used for developing the data modules on separate data servers, then this modeling paradigm cannot be addressed without relying on a global communication. The present embodiment addresses how much communication is needed, and what is the nature of the information that is communicated.

Since the data modules are developed separately from the regression application, and since kMER is used for it, an elegant solution can be found that minimizes the need for global communication. This results in a new procedure, compatible with the one introduced in the previous subsection. For the sake of exposition, it is assumed that the subspace data sets are stored onto separate data servers. However, it is clear that they can be grouped onto one or more servers. The overall scheme is shown in Fig. 12.

For each data server, a data module is developed with kMER using the

subspace vectors that are locally available. It is assumed that the desired output values (scalars) are available on each data server. m (level 1) regression modules are then locally developed, as explained in section 3.2.1, thus, using the subspace vectors as input vectors. The outputs of the m regression modules still have to be integrated into one global regression result. This is done by applying backfitting (see section 3.2.1). In order to generate an initial regression estimate, from which backfitting can start, the outputs produced by each of the m regression modules are averaged:

$$O(\mathbf{x}^{\mu}) \doteq \hat{y}^{\mu} = \sum_{k=1}^{m} \frac{1}{m} O_k(\mathbf{a}_k \mathbf{x}^{\mu T}), \tag{30}$$

with $\mathbf{a}_k$ a vector, the components of which are 1 when the corresponding input dimension is coded for by the kth regression model, else it is 0. Note the resemblance with projection pursuit regression (PPR), except that here the $\mathbf{a}_k$ are not subject to optimization.

Next, the following algorithm is considered that is applied to a training set that is allowed to be different from the one used for training the m regression modules.

Backfitting initialization

For each module j, the module's regression output $O_j(\mathbf{a}_j \mathbf{x}^{\mu T})$ is determined for each input vector of the training set. These outputs (*i.e.*, scalars) are then communicated to all other data servers. Hence, in this way, each data server disposes of the regression outputs produced by all regression modules for all the input vectors used for training. A module index, i, is intruduced and is put $i \leftarrow 1$. Let ε be a preset level of training error. The total regression error (TRE) is defined as follows:

$$\mathrm{TRE} = \frac{1}{M} \sum_{\mu=1}^{M} \left(y^{\mu} - \sum_{k=1}^{m} \frac{1}{m} O_k(\mathbf{a}_k \mathbf{x}^{mT}) \right), \tag{31}$$

which is preferably determined for a separate test set, rather than the training set. The TRE is communicated to regression module 1.

do until TRE $< \varepsilon$ or until maximum number of iterations is reached

for ($i \Leftarrow 1$; $i \leq m$; $i \Leftarrow i + 1$)

backfitting run i

Module i's regression parameters are adapted so as to reduce the total regression

error (gradient descent step, in batch mode); the parameters of all regression modules are kept constant. The regression outputs of module i are determined for each subspace input vector of the training set and communicated to the next module on which a learning step is going to be performed, module $i + 1$. The module index is incremented, $i \leftarrow i + 1$, and another backfitting run is performed.

In summary, a continued cyclical update of the modules is carried out until the total regression error is small enough or a more elaborate stop criterion has been satisfied (*e.g.*, either TRE $< \varepsilon$ or the maximum number of iterations has been reached).

It should be noted that only a restricted amount of information is exchanged between the data servers, and only when needed during learning. Furthermore, only model outputs are exchanged and not input data, which would require more communication effort and could, *e.g.*, raise concerns about ownership and confidentiality of the data, etc.

Finally, when backfitting has stopped, the regression modules can be used as follows. First, the subspace input vectors are applied to each regression module, then the corresponding regression module outputs O_k are determined and communicated to the level 2 regression module which, in turn, calculates the final regression result:

$$O(\mathbf{x}^{\mu}) \doteq \hat{y}^{\mu} = \sum_{k=1}^{m} \frac{1}{m} O_k(\mathbf{a}_k \mathbf{x}^{\mu T}). \qquad (32)$$

4.3.2. Vectorial regression task

The previous embodiment can be easily extended to vectorial regression, thus with d_y-dimensional desired output vectors, rather than scalars: d_y regression models are developed, one for each output vector component. All training processes can run in parallel.

4.4. Subsets of subspaces – mixed case

The case where the data modules are trained on subsets that consist of mixtures of input space vectors as well as subspace vectors is now considered, thus with missing input vector components (see Fig. 8C). There are two possibilities.

If there are plenty of input space vectors available for developing the data modules and/or the subspace vectors only have a limited number of missing vector components, then the presence of the missing vector components can be ignored and

the strategy mentioned under section 4.2 can be applied: m regression models are developed using the input space vectors as well as the subspace input vectors (Fig. 13). For the subspace input vectors the data completion procedure explained in section 3.1.2 is used.

In the opposite case, if there are plenty of subspace vectors available, then the strategy mentioned under section 4.3.1 should be considered, and the data modules should be developed for each subspace separately, thereby ignoring the additional vector components that some of the data vectors might have (Fig. 14).

The first strategy is clearly an heuristic one: its success will critically depend on the ratio between the number of complete and incomplete input vectors, and the number of missing input vector components. The second strategy is in principle correct but it requires much more data communication than in the previous case.

Finally, there is also the general case where the data vector dimensionalities vary in the training subsets. The subspace dimensionality that is used for each data module might correspond to the set of vector components that are in common to the training subset or, if this is too extreme, a common set of vector components can be chosen or determined, and data completion can be performed on the missing vector components and the ones that do not belong to this common set can be simply ignored. Evidently, such an approach is also an heuristic one.

4.4.1. Vectorial regression task

Again, vectorial regression application can be performed by developing d_y regression models in parallel.

5. Utility

The systems and methods described above may be used to model a physical system using a plurality of data models. A regression analysis may then be applied to the modeled system using the plurality of data models. A physical parameter of the modeled system may be changed, optimized, or controlled in accordance with the output of a regression model determined in accordance with the present invention. The output of the regression model is tangible, concrete and useful.

6. Additional References

Friedman, J.H., and Silverman, B.W. (1989). Flexible parsimonious smoothing and additive modeling. *Technometrics*, **31**(1), 3-21.

Moody, J., and Darken, C. (1988). Learning with localized receptive fields. *Proc. 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton and T. Sejnowski (Eds.), pp. 133-143, San Mateo: Morgan Kaufmann.

Poggio, T., and Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, **247**, 978-982.

Specht, D.F. (1991). A general regression neural network. *IEEE Trans. on Neural Networks*, **2**(6), 568-576.

CLAIMS

1. A computer implemented regression method for carrying out a regression application on a plurality of input data subsets of an input data signal, comprising the steps of:

developing for each of the input data subsets a data model independently of the regression application to form a set of data models;

generating a regression model for the input data signal using the set of data models.

2. The method according to claim 1, wherein the developing step includes a map developing step in which an unsupervised, self-organizing, kernel-based, topographic map is developed from each input data subset by maximizing the entropy of the map's output, the topographic map being executed by a neural network; and
during the development step receptive fields of the topographic map are truncated to receptive field regions, the receptive field regions being independently scaleable in the space of the input data signal.

3. The method according to claim 2, wherein the map is equiprobabilistic.

4. The method according to claim 2 or 3, wherein at least some of the receptive field regions overlap.

5. The method according to claim 4, wherein the receptive fields are defined by the same kernel function.

6. The method according to claim 5, wherein the receptive field regions are hyper-spheroidal.

7. The method according to any of the claims 1 to 6, wherein each data model comprises data points and a real function and the map developing step further comprises estimating a data density for each data point of a data model and the generating step includes the step of selecting a data point for the regression step based on the data density for that point.

8. The method according to any of the claims 2 to 7, wherein the developing step comprises adapting the center location and the extent of each receptive field region to the input data density of the part of the input data signal data which activates the kernel thereof.

9. The method according to claim 1, wherein the developing step includes use of any of the following: self-organizing maps (SOM), topographic maps, constrained topological mapping (CTM), transformation methods such as principal component analysis (PCA), independent component analysis (ICA), multidimensional scaling or similar methods, especially unsupervised methods.

10. The method according to any previous claim wherein the generating step includes any of the following: multilayer perceptrons, (smoothing) splines, support vector machines (SVM), Radial Basis Function (RBF) networks, general regression neural network (GRNN), constrained topological mapping (CTM), Generative Topographic Map (GTM), the mixture of experts network.

11. The method according to any previous claim, wherein each data record in the input data signal is contained within one input data subset.

12. The method according to any of the claims 1 to 10, wherein each data record in the input data signal is distributed over the input data subsets.

13. The method according to any of the claims 1 to 10 wherein some data records in the input data signal are contained within one input data subset and some data records are spread over some of the data subsets.

14.The method according to any previous claim wherein the step of generating the regression model comprises generating a regression sub-model for each of the data models, and combining the regression sub-models.

15. A computer based system for executing a regression application on a plurality of

input data subsets of an input data signal, comprising:

means for developing for each of the input data subsets a data model independently of the regression application to form a set of data models; and

means for generating a regression model for the input data signal using the set of data models.

16. The computer based system according to claim 15, wherein the means for developing includes a neural network in which an unsupervised, self-organizing, kernel-based, topographic map is developed from each input data subset by maximizing the entropy of the map's output, and

means for truncating receptive fields of the topographic map to receptive field regions, the receptive field regions being independently scaleable in the space of the input data signal.

17. The computer based system of claim 15 or 16 the means for generating the regression model comprises means for generating a regression sub-model for each of the data models, and means for combining the regression sub-models. 17. The computer based system of any of claims 14 to 16, further comprising means for generating a data density model from each data model.

18. The computer based system according to claim 17, wherein each data model comprises data points and a real function, further comprising decision means for selecting which data point of each data model is used for the generation of the regression model based on the output of the means for generating the data density model for that data point.

19. A computer program product for causing a processing engine to execute any of the methods according to claims 1 to 14.

20. A data carrier storing code segments of a computer program for executing any of the methods according to claims 1 to 14.

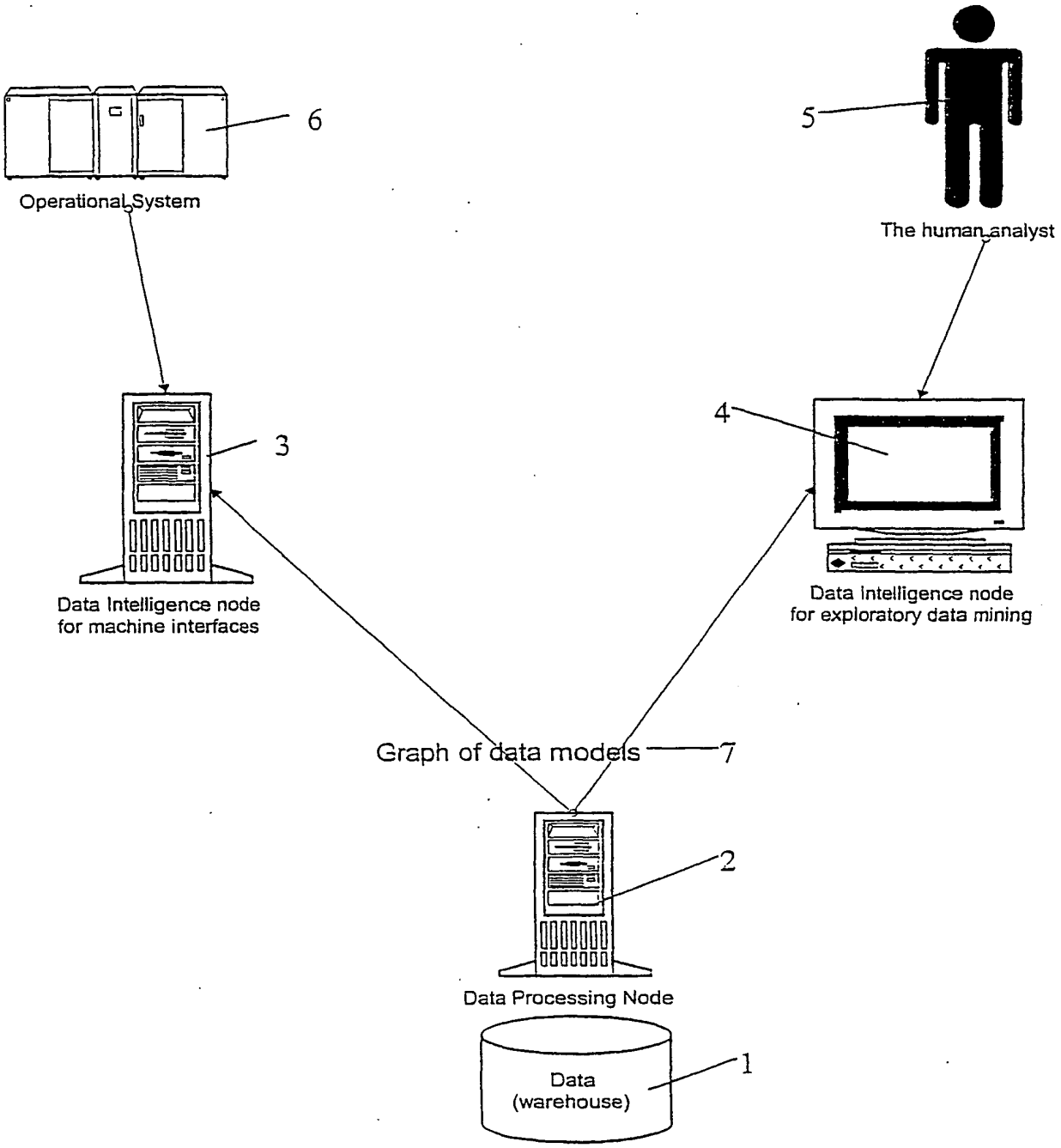


Fig. 1

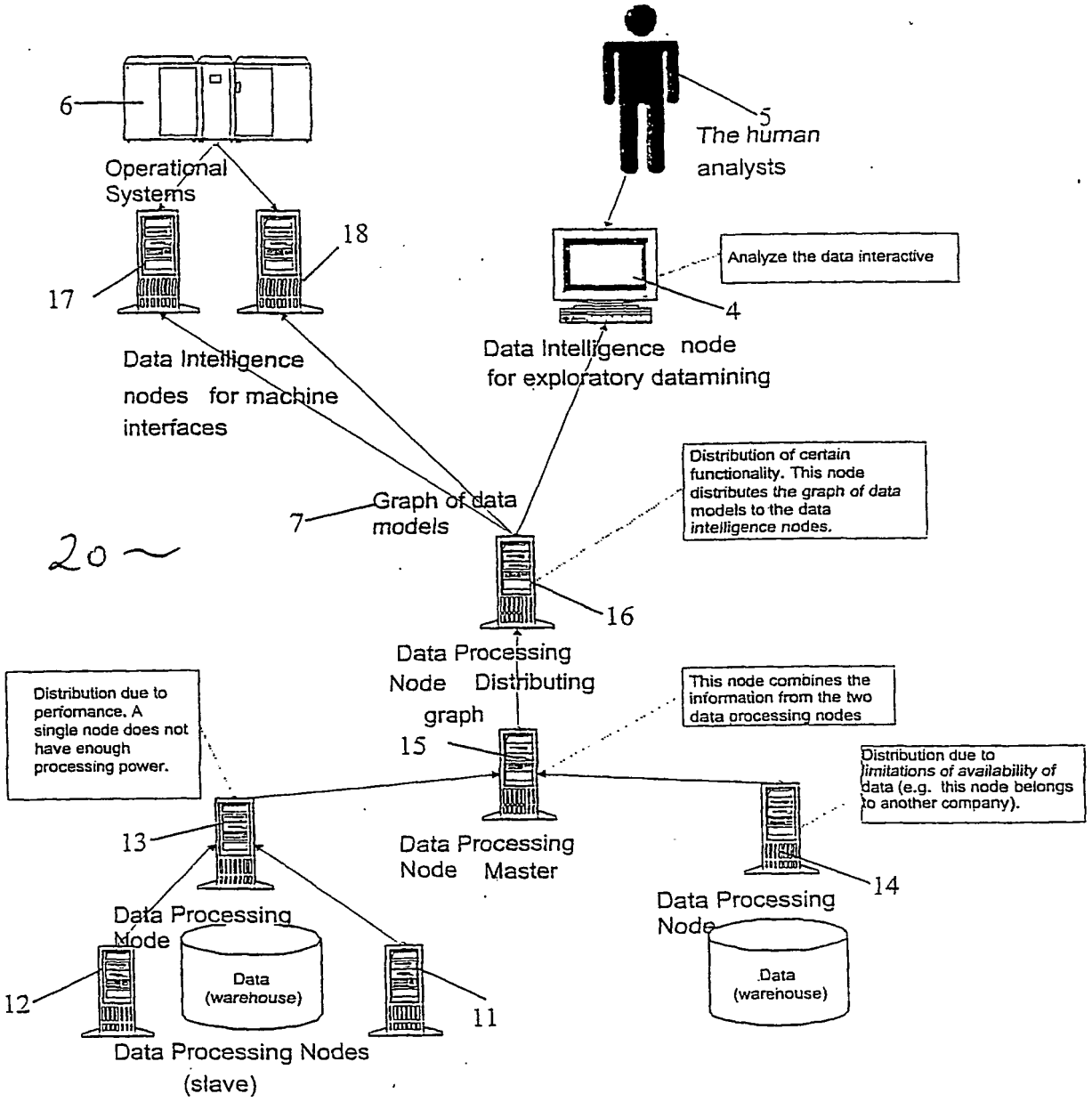


Fig. 2

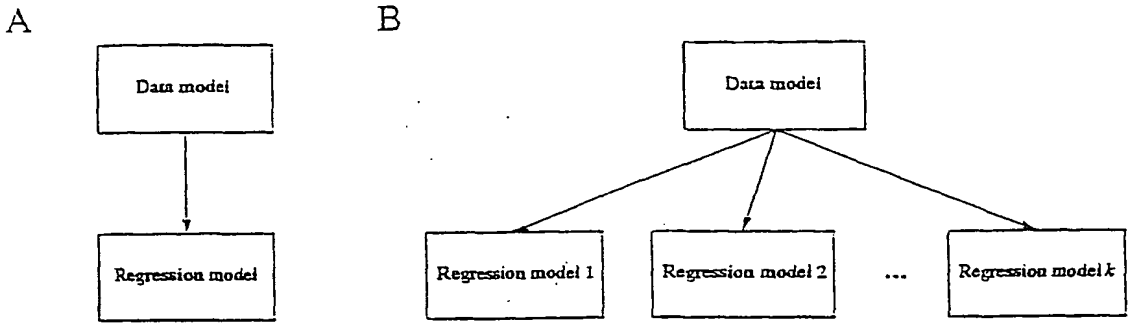


Fig. 3

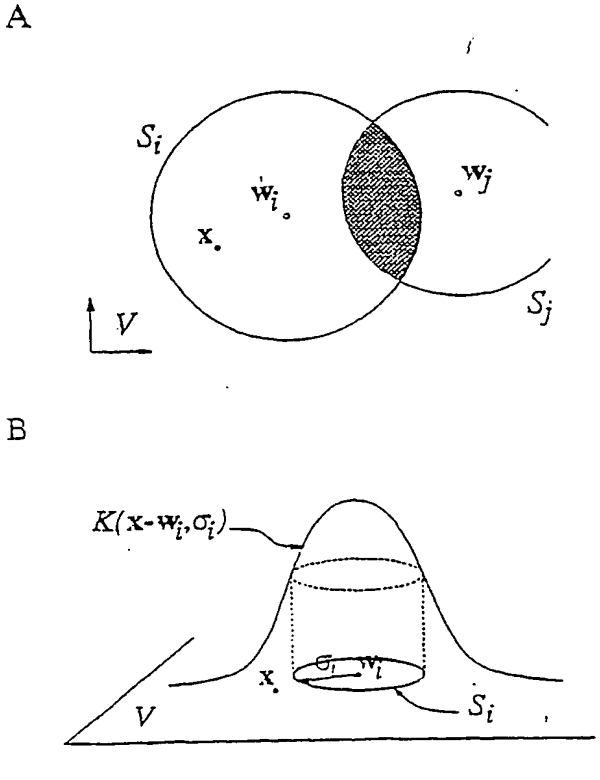


Fig. 4

$Lattice_{ij}$ = *Euclidean distance* $(neuron_i, neuron_j)_{in\ lattice\ coordinates}$

Missing values for input vector components v_j are marked as "×"

for$(t \Leftarrow 1; t \leq t_{max}; t \Leftarrow t + 1)$

/* *Initialize* Δw_{ij} *and* $\Delta\sigma_i$ *to zero* */

for$(i \Leftarrow 1; i \leq N; i \Leftarrow i + 1)$

for$(j \Leftarrow 1; j \leq d; j \Leftarrow j + 1)$

$\Delta w_{ij} \Leftarrow 0$

$\Delta\sigma_i \Leftarrow 0$

$$\sigma_\Lambda \Leftarrow \sigma_{\Lambda 0} \cdot exp\left(-2\frac{\sigma_{\Lambda 0} \cdot t}{t_{max}}\right)$$

for$(\mu \Leftarrow 1; \mu \leq M; \mu \Leftarrow \mu + 1)$

/* *Compute discrepancy (v, w)* */

/* *Determine closest neuron and active neurons* */

$i^* \Leftarrow 1$

for$(i \Leftarrow 1; count \Leftarrow 0; i \leq N; i \Leftarrow i + 1)$

for$(j \Leftarrow 1, D2_i \Leftarrow 0; j \leq d; j \Leftarrow j + 1)$

if$(v_j^\mu \neq \times)$ /* *Not a missing entry* */

$D_{ij} \Leftarrow v_j^\mu - w_{ij}$

$D2_i \Leftarrow D2_i + D_{ij}^2$

if$(D2_i < D2_{i^*})$

$i^* \Leftarrow i$

if$(D2_i < \sigma_i^2)$

$CM_i \Leftarrow 1$

$count \Leftarrow count + 1$

else

$CM_i \Leftarrow 0$

/* *The rest of the algorithm stays the same, as in (Van Hulle, 1999a) and is given on the next page* */

Fig. 5

$Lattice_{ij}$ = *Euclidean distance* $(neuron_i, neuron_j)_{in\ lattice\ coordinates}$

for$(t \Leftarrow 1; t \leq t_{max}; t \Leftarrow t + 1)$

/* *Initialize* Δw_{ij} *and* $\Delta\sigma_i$ *to zero* */

for$(i \Leftarrow 1; i \leq N; i \Leftarrow i + 1)$

for$(j \Leftarrow 1; j \leq d; j \Leftarrow j + 1)$

$\Delta w_{ij} \Leftarrow 0$

$\Delta\sigma_i \Leftarrow 0$

$\sigma_\Lambda \Leftarrow \sigma_{\Lambda 0} \cdot exp^{-2\frac{\sigma_{\Lambda 0} \cdot t}{t_{max}}}$

for$(\mu \Leftarrow 1; \mu \leq M; \mu \Leftarrow \mu + 1)$

select v from {distribution set}

/**Compute discrepancy (v, w) and determine closest neuron and active neurons**/

for$(i \Leftarrow 1; x_1 \Leftarrow \infty, count \Leftarrow 0; i \leq N; i \Leftarrow i + 1)$

for$(j \Leftarrow 1, D2_i \Leftarrow 0; j \leq d; j \Leftarrow j + 1)$

$D_{ij} \Leftarrow v_j^\mu - w_{ij}$

$D2_i \Leftarrow D2_i + D_{ij}^2$

if$(D2_i < x_1)$

$x_1 \Leftarrow D2_i$

$winner \Leftarrow i$

if$(D2_i < \sigma_i^2)$

$CM_i \Leftarrow 1$

$count \Leftarrow count + 1$

else

$CM_i \Leftarrow 0$

if$(count == 0)$ /* *No neurons active* */

$count \Leftarrow 1$

$CM_{winner} \Leftarrow 1$ /* *Closest neuron made active* */

Fig. 5 CONT.

$$\Delta\sigma_{winner} \Leftarrow \Delta\sigma_{winner} + \frac{\rho}{N} + 1 \qquad \text{/* Adapt radius */}$$

/* *Compute partial update* */

for$(i \Leftarrow 1;\ i \le N;\ i \Leftarrow i + 1)$

if$(CM_i == 0)$ /* *Non-active neurons* */

$$D3 \Leftarrow \frac{1}{count} \cdot exp\left(\frac{Lattice^2_{winner,i}}{2\sigma^2_\Lambda}\right)$$

for$(j \Leftarrow 1; j \le d; j \Leftarrow j + 1)$

$$\Delta w_{ij} \Leftarrow \Delta w_{ij} + D3 \cdot sign(D_{ij})$$

$$\Delta\sigma_i \Leftarrow \Delta\sigma_i + \frac{\rho}{N}$$

else /* *Active neurons* */

for$(j \Leftarrow 1; j \le d; j \Leftarrow j + 1)$

$$\Delta w_{ij} \Leftarrow \Delta w_{ij} + \frac{sign(D_{ij})}{count}$$

$$\Delta\sigma_i \Leftarrow \Delta\sigma_i - 1$$

/* *Batch update of weight vectors* */

for$(i \Leftarrow 1;\ i \le N;\ i \Leftarrow i + 1)$

for$(j \Leftarrow 1; j \le d; j \Leftarrow j + 1)$

$$w_{ij} \Leftarrow w_{ij} + \eta \cdot \Delta w_{ij}$$

$$\sigma_i \Leftarrow \sigma_i + \eta \cdot \Delta\sigma_i$$

Fig. 5 cont.

Given: input vector of which some components v_j are missing "×"

Find: estimates for missing vector components

```
/* Compute discrepancy (v, w) */
/* Determine closest neuron */
i* ⇐ 1
for(i ⇐ 1; count ⇐ 0; i ≤ N; i ⇐ i + 1)
   for(j ⇐ 1, D2_i ⇐ 0; j ≤ d; j ⇐ j + 1)
      if(v_j^μ ≠ ×) /* Not a missing entry */
         D_ij ⇐ v_j^μ − w_ij
         D2_i ⇐ D2_i + D_ij^2
   if(D2_i < D2_i*)
      i* ⇐ i

/* Fill-in missing input vector components */
for(j ⇐ 1; j ≤ d; j ⇐ j + 1)
   if(v_j^μ == ×) /* Missing entry */
      v_j^μ ⇐ w_i*j
```

Figure 6

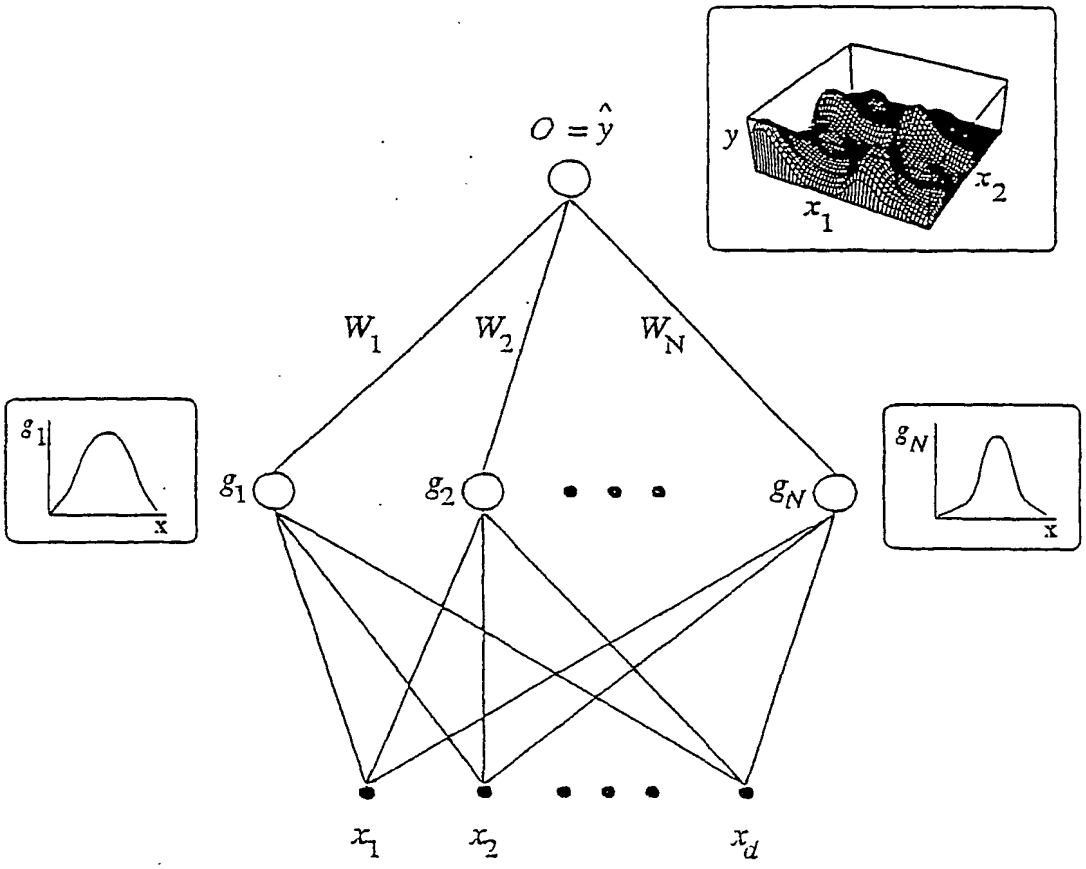
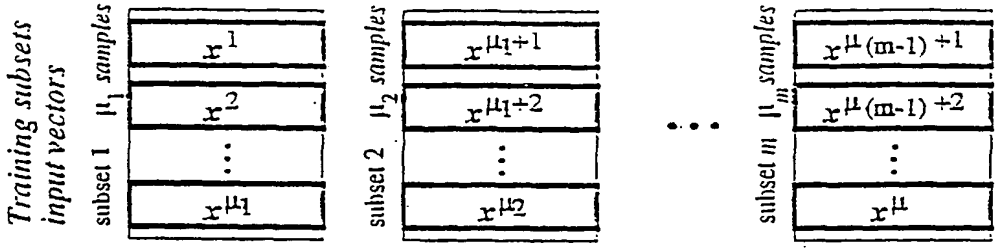
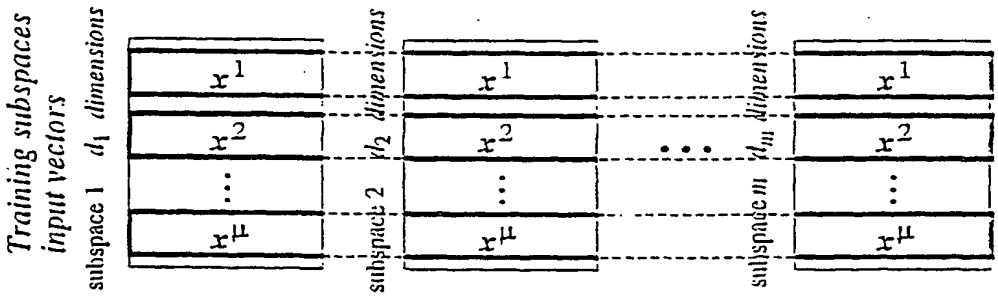


Fig. 7

A



B



C

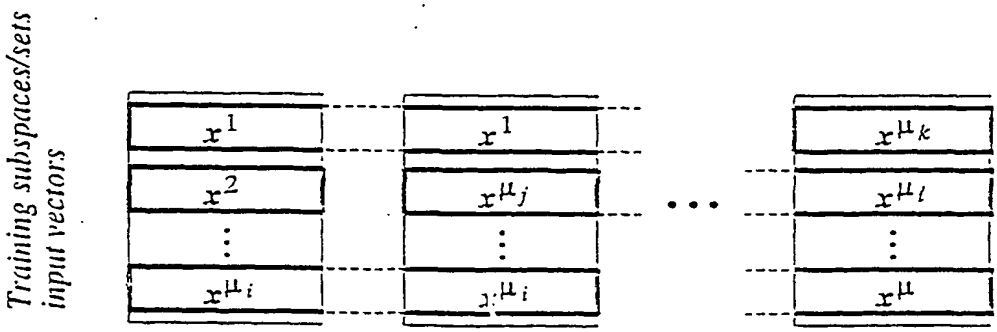


Fig. 8

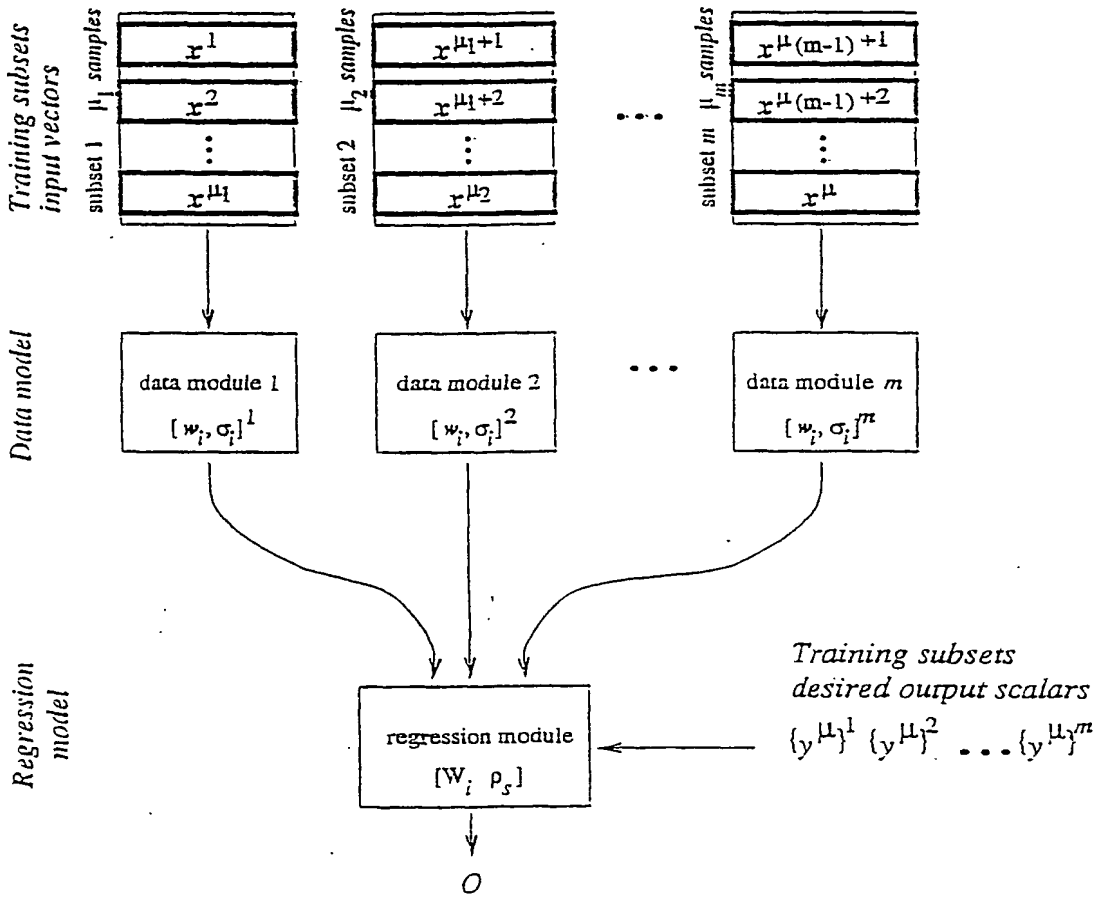


Fig. 9

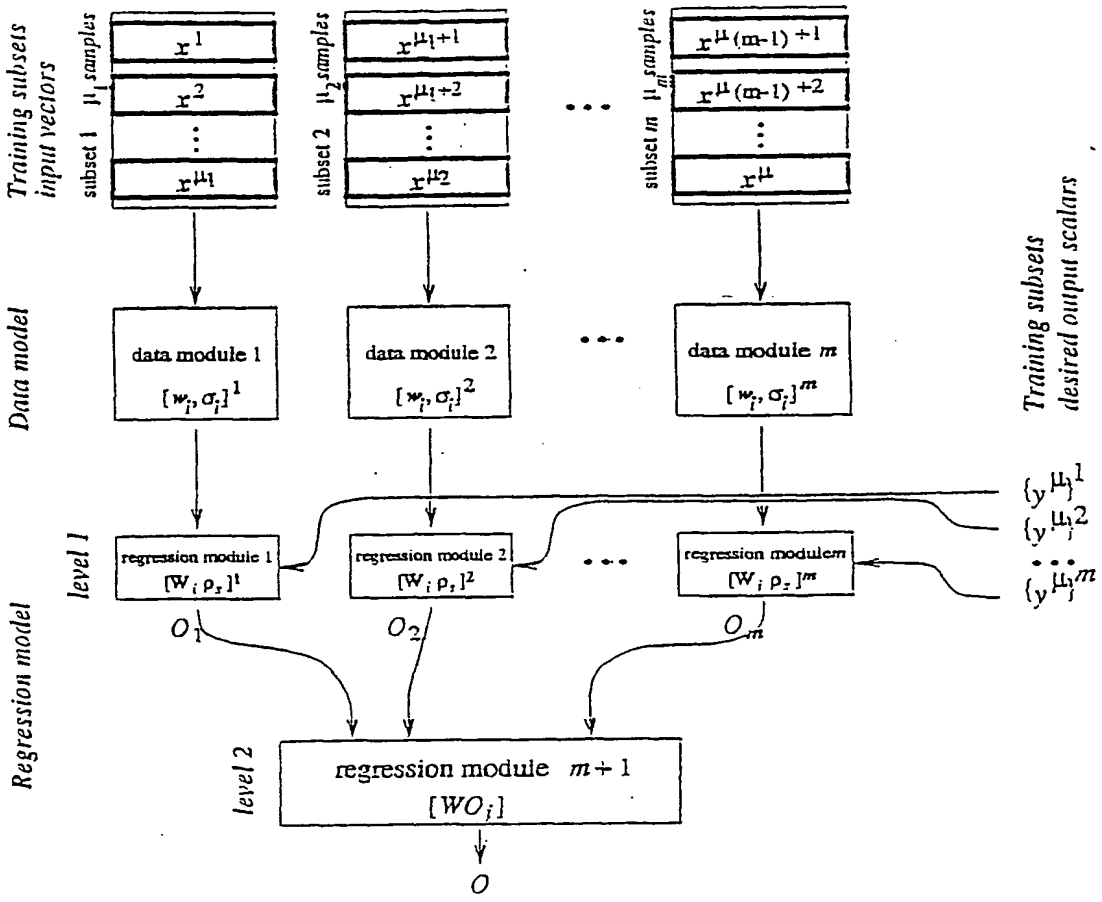


Fig. 10

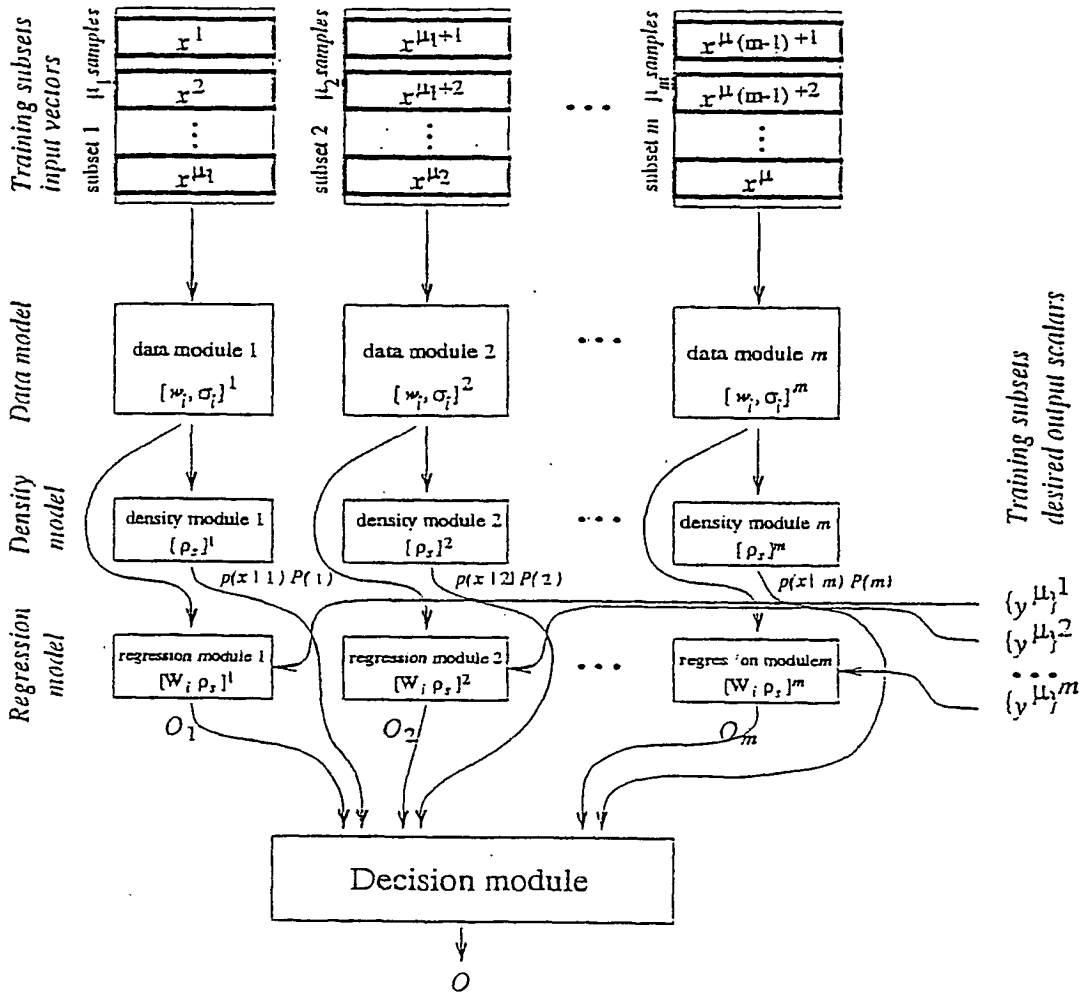


Fig. 11

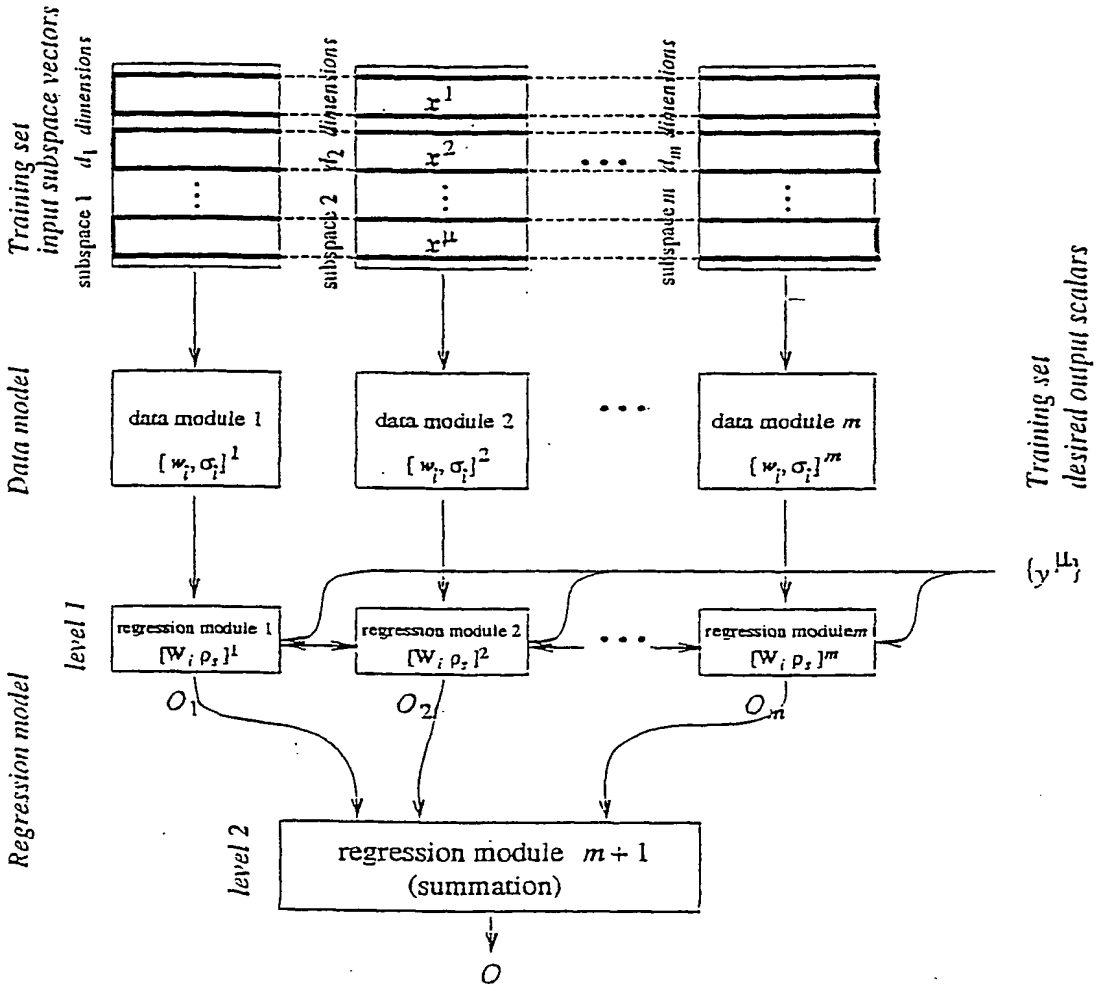


Fig. 12

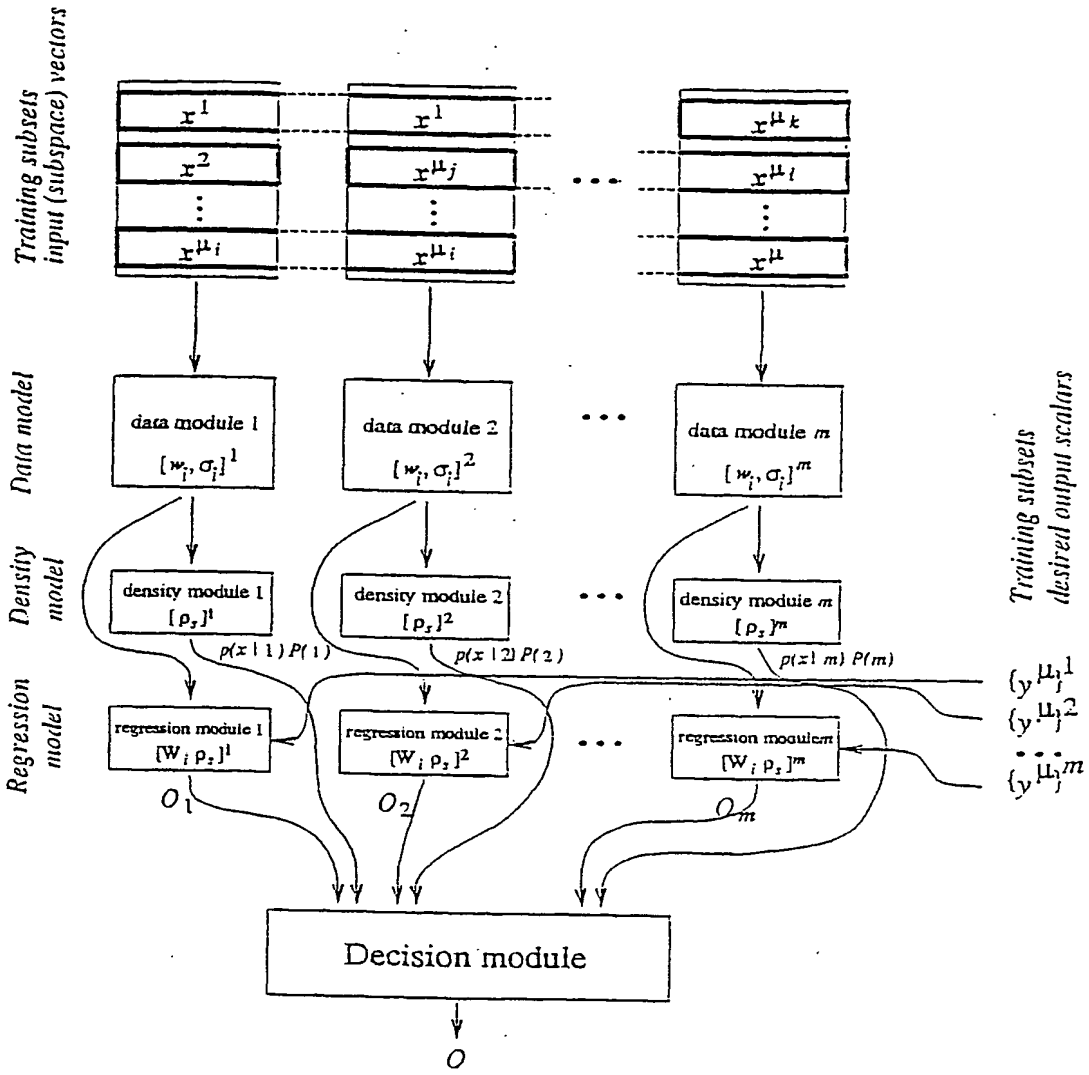


Fig. 13

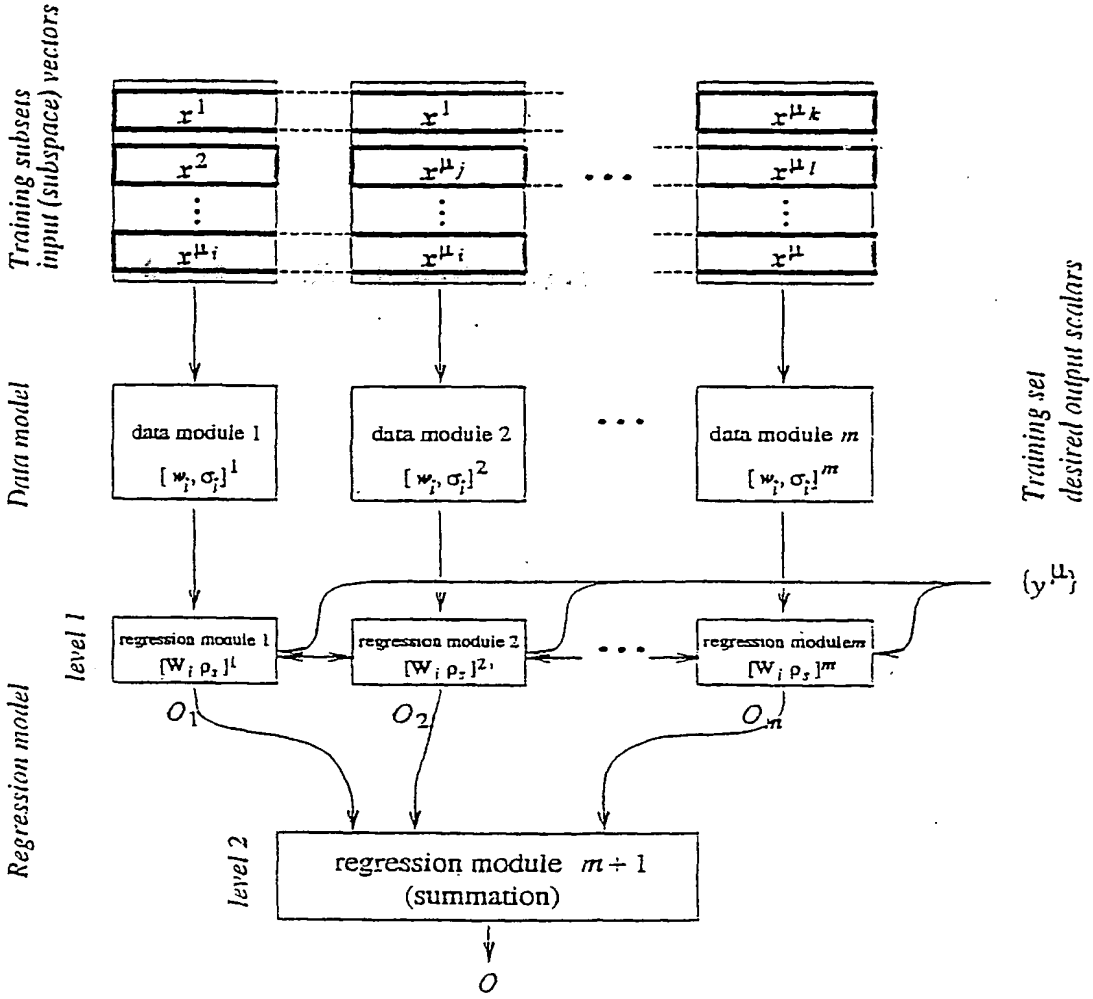


Fig. 14

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 October 2001 (25.10.2001)

PCT

(10) International Publication Number
WO 01/80176 A3

(51) International Patent Classification[7]: G06N 3/02

(21) International Application Number: PCT/BE01/00065

(22) International Filing Date: 13 April 2001 (13.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0008985.4 13 April 2000 (13.04.2000) GB
PCT/BE00/00099 30 August 2000 (30.08.2000) BE

(71) Applicant *(for all designated States except US)*: SYNES NV [BE/BE]. Technologielaan 11, B-3000 Leuven (BE).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: VAN HULLE, Marc [BE/BE]. Hoefstadstraat 72, B-3600 Genk (BE).

(74) Agents: BIRD, William, E. et al.; Bird Goën & Co, Vilvoordsebaan 92, B-3020 Winksele (BE).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

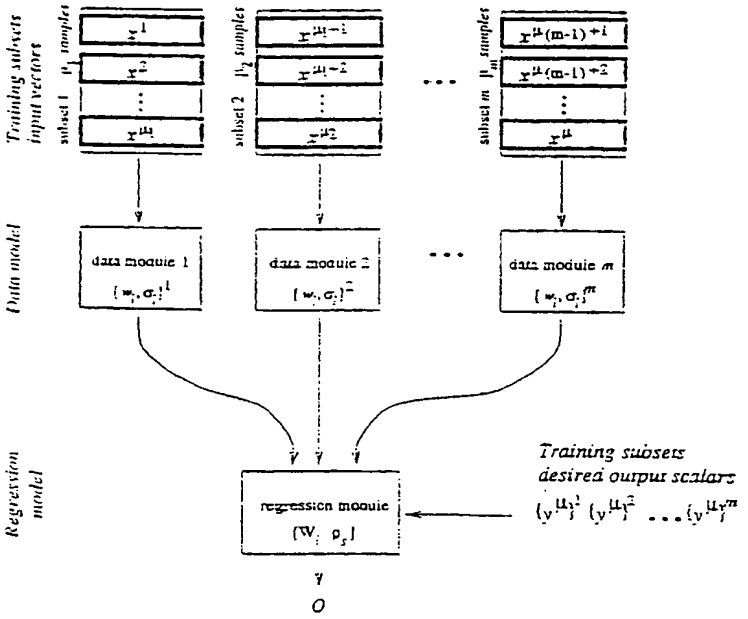
Declaration under Rule 4.17:
— *of inventorship (Rule 4.17(iv)) for US only*

Published:
— *with international search report*

(88) Date of publication of the international search report:
11 April 2002

[Continued on next page]

(54) Title: METHODS AND SYSTEMS FOR REGRESSION ANALYSIS OF MULTIDIMENSIONAL DATA SETS



(57) Abstract: The present invention may provide a method and system for distributed regression modeling. An initial or source data set is in a partitioned form, i.e. in one or more subsets, and data and regression modules are developed separately for these data subsets. The subsets can consist of either complete data vectors or incomplete data vectors, or of a mixture of the two. The data vectors may be incomplete, for example, since they contain missing vector components.

WO 01/80176 A3

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

Inter. Application No
PCT/BE 01/00065

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06N3/02

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
INSPEC, EPO-Internal, WPI Data, PAJ, IBM-TDB, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropnate, of the relevant passages	Relevant to claim No.
X	VAN HULLE M M: "Nonparametric regression modeling with topographic maps as a basis for lossy image compression" NEURAL NETWORKS FOR SIGNAL PROCESSING VII. PROCEEDINGS OF THE 1997 IEEE SIGNAL PROCESSING SOCIETY WORKSHOP, AMELIA ISLAND, FL, USA 24-26 SEPT. 1997, pages 4-13, XP002183198 1997, New York, NY, USA, IEEE, USA ISBN: 0-7803-4256-9	1,2,9, 15,16, 19,20
Y	page 4, line 1 -page 12, line 21	3-5,10
A	figures 1-3	6-8, 11-14, 17,18

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 November 2001	30/11/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Schenkels, P

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	VAN HULLE M M: "Faithful representations with topographic maps" NEURAL NETWORKS, ELSEVIER SCIENCE PUBLISHERS, BARKING, GB, vol. 12, no. 6, July 1999 (1999-07), pages 803-823, XP004174077 ISSN: 0893-6080 page 803, line 1 -page 811, left-hand column, line 7; figures 1-4 ---	3-5,10
A	VAN HULLE M M: "Nonparametric regression analysis achieved with topographic maps developed in combination with projection pursuit learning: an application to density estimation and adaptive filtering of grey-scale images" IEEE TRANSACTIONS ON SIGNAL PROCESSING, NOV. 1997, IEEE, USA, vol. 45, no. 11, pages 2663-2672, XP002183199 ISSN: 1053-587X abstract ---	1,15
A	US 5 701 395 A (STUCKMAN BRUCE EDWARD ET AL) 23 December 1997 (1997-12-23) abstract -----	1,15

1